Integration Panel:

Root Zone Label Generation Rules — LGR-3 Overview and Summary

REVISION – July 10, 2019

Table of Contents

1 Overview **5**

1.1 Root Zone Label Generation Rules (LGR-3) Files *5*

2 Process of Integration **7**

2.1 Overview *7*

2.2 Proposals Submitted *9*

2.3 Review of Proposals *11*

2.3.1 General Notes on the Proposal Review 11

2.3.2 Arabic LGR Proposal Review 12

2.3.3 Armenian LGR Proposal Review 12

2.3.4 Cyrillic LGR Proposal Review 12

2.3.5 Devanagari LGR Proposal Review 13

2.3.6 Ethiopic LGR Proposal Review 14

2.3.7 Georgian LGR Proposal Review 14

2.3.8 Gujarati LGR Proposal Review 14

2.3.9 Gurmukhi LGR Proposal Review 15

2.3.10 Hebrew LGR Proposal Review 15

2.3.11 Kannada LGR Proposal Review 16

2.3.12 Khmer LGR Proposal Review 16

2.3.13 Lao LGR Proposal Review 17

2.3.14 Malayalam LGR Proposal Review 17

2.3.15 Oriya LGR Proposal Review 18

2.3.16 Sinhala LGR Proposal Review 19

2.3.17 Tamil LGR Proposal Review 20

2.3.18 Telugu LGR Proposal Review 20

2.3.19 Thai LGR Proposal Review 21

3 Integration and Contents of LGR-3 **21**

3.1 General Notes *21*

3.1.1 Summary 22

3.2 Merged LGR (Common) *23*

3.2.1 Repertoire 23

3.2.2 Variants 23
3.2.3 Character Classes 24
3.2.4 Whole-Label Evaluations (WLE) Rules 24
3.3 Arabic Element LGR *25*
3.3.1 Repertoire for Arabic 25
3.3.2 Variants for Arabic 25
3.3.3 Whole-Label Evaluation Rules for Arabic 25
3.3.4 Default Whole-Label Evaluation Rules 25
3.4 Devanagari Element LGR *26*
3.4.1 Repertoire for Devanagari 26
3.4.2 Variants for Devanagari 26
3.4.3 Whole-Label Evaluation Rules for Devanagari 26
3.4.4 Default Whole-Label Evaluation Rules 26
3.5 Ethiopic Element LGR *26*
3.5.1 Repertoire for Ethiopic 26
3.5.2 Variants for Ethiopic 27
3.5.3 Whole-Label Evaluation Rules for Ethiopic 27
3.5.4 Default Whole-Label Evaluation Rules 27
3.6 Georgian Element LGR *27*
3.6.1 Repertoire for Georgian 27
3.6.2 Variants for Georgian 27
3.6.3 Whole-Label Evaluation Rules for Georgian 27
3.6.4 Default Whole-Label Evaluation Rules 27
3.7 Gujarati Element LGR *27*
3.7.1 Repertoire for Gujarati 27
3.7.2 Variants for Gujarati 28
3.7.3 Whole-Label Evaluation Rules for Gujarati 28
3.7.4 Default Whole-Label Evaluation Rules 28
3.8 Gurmukhi Element LGR *28*
3.8.1 Repertoire for Gurmukhi 28
3.8.2 Variants for Gurmukhi 28
3.8.3 Whole-Label Evaluation Rules for Gurmukhi 28
3.8.4 Default Whole-Label Evaluation Rules 28
3.9 Hebrew Element LGR *29*
3.9.1 Repertoire for Hebrew 29
3.9.2 Variants for Hebrew 29
3.9.3 Whole-Label Evaluation Rules for Hebrew 29
3.9.4 Defaults Whole-Label Evaluation Rules 29
3.10 Kannada Element LGR *29*
3.10.1 Repertoire for Kannada 29
3.10.2 Variants for Kannada 29
3.10.3 Whole-Label Evaluation Rules for Kannada 29

3.10.4 Default Whole-Label Evaluation Rules 30
3.11 Khmer Element LGR *30*
3.11.1 Repertoire for Khmer 30
3.11.2 Variants for Khmer 30
3.11.3 Whole-Label Evaluation Rules for Khmer 30
3.11.4 Default Whole-Label Evaluation Rules 30
3.12 Lao Element LGR *30*
3.12.1 Repertoire for Lao 30
3.12.2 Variants for Lao 31
3.12.3 Whole-Label Evaluations Rules for Lao 31
3.12.4 Default Whole-Label Evaluation Rules 31
3.13 Malayalam Element LGR *31*
3.13.4 Default Whole-Label Evaluation Rules 32
3.14 Oriya (Odia) Element LGR *32*
3.14.1 Repertoire for Oriya 32
3.14.2 Variants for Oriya 32
3.14.3 Whole-Label Evaluation Rules for Oriya 33
3.14.4 Default Whole-Label Evaluation Rules 33
3.15 Sinhala Element LGR *33*
3.15.1 Repertoire for Sinhala 33
3.15.2 Variants for Sinhala 33
3.15.3 Whole-Label Evaluation Rules for Sinhala 33
3.15.4 Default Whole-Label Evaluation Rules 33
3.16 Tamil Element LGR *34*
3.16.1 Repertoire for Tamil 34
3.16.2 Variants for Tamil 34
3.16.3 Whole-Label Evaluation Rules for Tamil 34
3.16.4 Default Whole-Label Evaluation Rules 34
3.17 Telugu Element LGR *34*
3.17.1 Repertoire for Telugu 34
3.17.2 Variants for Telugu 35
3.17.3 Whole-Label Evaluation Rules for Telugu 35
3.17.4 Default Whole-Label Evaluation Rules 35
3.18 Thai Element LGR *35*
3.18.1 Repertoire for Thai 35
3.18.2 Variants for Thai 35
3.18.3 Whole-Label Evaluations Rules for Thai 35
3.18.4 Default Whole-Label Evaluation Rules 36
4 General Notes on the Root Zone LGR **36**
4.1 Rules *36*
4.2 Scripts *36*

4.3 Comprehensiveness and Staging *37*

5 Using the LGR **37**

5.1 Element LGRs *37*

5.2 Common LGR *37*

5.3 Other uses of the Common LGR *38*

5.4 Steps in Processing a Label *38*

5.5 Index Label Calculation *39*

5.5.1 Background 39

5.5.2 Transitivity of Code Point Variant Sets and Variant Label Sets 39

5.5.3 Requirements for Index Labels 40

5.5.4 Generating Index Labels 40

5.5.5 Impact on Root Zone LGR 41

6 Design Notes for the Root Zone LGR **41**

6.1 Reducing Complexity *41*

6.2 Limitations of the LGR *42*

6.2.1 Unicode Version 6.3.0 42

6.3 Cross-Script Variants and Security *42*

6.3.1 Related Scripts and Cross-Script Variants 43

6.3.2 Transitive Closure 44

6.4 Code Point Sequences *44*

6.4.1 Sequences and Context Rules 44

6.4.2 Sequences Defined For Use as Variants 44

6.5 Effective Null Variants *45*

6.6 Overlapped Variants *46*

7 Summary of Changes **48**

7.1 Changes by revision *48*

7.2 Code points by script *48*

8 Contributors **49**

8.1 Integration Panel Members *49*

8.2 Advisors *49*

8.3 Community Members *49*

8.4 ICANN Staff *49*

9 References **50**

1 Overview

This document describes the Label Generation Rules for the Root Zone (LGR) developed according to the "Procedure to Develop and Maintain the Label Generation Rules for the Root Zone in Respect of IDNA Labels" [Procedure]. The Procedure defines a two-stage process, in which community-based Generation Panels (GP) propose LGRs specific to a given script, which are then reviewed and integrated by the Integration Panel (IP). The result of the current round of this development work is the third version of the LGR (LGR-3), which is fully backwards compatible with [LGR-2] and [LGR-1].

The reader of this document is assumed to be familiar with the [Procedure][1], particularly the parts that describe the role of the IP and the tasks and expectations on the GPs.

The full content of LGR-3 is specified in a set of files as described in the next section.

1.1 Root Zone Label Generation Rules (LGR-3) Files

LGR-3 is provided as a collection of files that are self-contained and supersede the files from previous versions. This document (https://www.icann.org/sites/default/files/lgr/lgr-3-overview-10jul19-en.pdf) provides background on the content and development of this version of the LGR. It also provides additional guidance to potential users of the LGR.

Table 1. Merged (Common) and Element LGR files [XML – normative]

Script	File URL
Common	https://www.icann.org/sites/default/files/lgr/lgr-3-common-10jul19-en.xml
Arabic	https://www.icann.org/sites/default/files/lgr/lgr-3-arabic-script-10jul19-en.xml
Devanagari	https://www.icann.org/sites/default/files/lgr/lgr-3-devanagari-script-10jul19-en.xml
Ethiopic	https://www.icann.org/sites/default/files/lgr/lgr-3-ethiopic-script-10jul19-en.xml
Georgian	https://www.icann.org/sites/default/files/lgr/lgr-3-georgian-script-10jul19-en.xml
Gujarati	https://www.icann.org/sites/default/files/lgr/lgr-3-gujarati-script-10jul19-en.xml
Gurmukhi	https://www.icann.org/sites/default/files/lgr/lgr-3-gurmukhi-script-10jul19-en.xml
Hebrew	https://www.icann.org/sites/default/files/lgr/lgr-3-hebrew-script-10jul19-en.xml
Kannada	https://www.icann.org/sites/default/files/lgr/lgr-3-kannada-script-10jul19-en.xml
Khmer	https://www.icann.org/sites/default/files/lgr/lgr-3-khmer-script-10jul19-en.xml
Lao	https://www.icann.org/sites/default/files/lgr/lgr-3-lao-script-10jul19-en.xml
Malayalam	https://www.icann.org/sites/default/files/lgr/lgr-3-malayalam-script-10jul19-en.xml
Oriya[2]	https://www.icann.org/sites/default/files/lgr/lgr-3-oriya-script-10jul19-en.xml
Sinhala	https://www.icann.org/sites/default/files/lgr/lgr-3-sinhala-script-10jul19-en.xml
Tamil	https://www.icann.org/sites/default/files/lgr/lgr-3-tamil-script-10jul19-en.xml
Telugu	https://www.icann.org/sites/default/files/lgr/lgr-3-telugu-script-10jul19-en.xml
Thai	https://www.icann.org/sites/default/files/lgr/lgr-3-thai-script-10jul19-en.xml

[1] References to documents cited are provided at the end.

[2] The Root Zone LGR uses the naming conventions from [ISO 15924] for script names. For general use, the name "Odia" is used for this script.

The normative definition of LGR-3 is provided as a set of XML files, consisting of one merged file, named “common”, and one XML file per script, as shown in Table 1.

The Label Generation rules are expressed using a standard format defined in "Representing Label Generation Rulesets in XML" [RFC7940]. The remainder of this document assumes that the reader is at least familiar with some of the general concepts presented in that RFC.

The common LGR consists of a list of code points or sequences defining the merged repertoire as well as a set of mappings providing the variant relations between these repertoire items. In addition, the file contains a merged set of Whole-Label Evaluation (WLE) rules for the root zone. Each code point in the file is annotated with the Unicode version in which it was first assigned, and the scripts in which it is used.

Each of the script-specific files contains all the Label Generation Rules applicable to labels from that script, and only those rules. These files are called Element LGRs. Each file contains a description, a repertoire with optional variants, and WLE Rules, as well as detailed references that link each included code point to a reference that provides data justifying that code point’s inclusion.

Table 2. Merged (Common) and Element LGR files [HTML – non-normative]

Script	File URL
Common	https://www.icann.org/sites/default/files/lgr/lgr-3-common-10jul19-en.html
Arabic	https://www.icann.org/sites/default/files/lgr/lgr-3-arabic-script-10jul19-en.html
Devanagari	https://www.icann.org/sites/default/files/lgr/lgr-3-devanagari-script-10jul19-en.html
Ethiopic	https://www.icann.org/sites/default/files/lgr/lgr-3-ethiopic-script-10jul19-en.html
Georgian	https://www.icann.org/sites/default/files/lgr/lgr-3-georgian-script-10jul19-en.html
Gujarati	https://www.icann.org/sites/default/files/lgr/lgr-3-gujarati-script-10jul19-en.html
Gurmukhi	https://www.icann.org/sites/default/files/lgr/lgr-3-gurmukhi-script-10jul19-en.html
Hebrew	https://www.icann.org/sites/default/files/lgr/lgr-3-hebrew-script-10jul19-en.html
Kannada	https://www.icann.org/sites/default/files/lgr/lgr-3-kannada-script-10jul19-en.html
Khmer	https://www.icann.org/sites/default/files/lgr/lgr-3-khmer-script-10jul19-en.html
Lao	https://www.icann.org/sites/default/files/lgr/lgr-3-lao-script-10jul19-en.html
Malayalam	https://www.icann.org/sites/default/files/lgr/lgr-3-malayalam-script-10jul19-en.html
Oriya	https://www.icann.org/sites/default/files/lgr/lgr-3-oriya-script-10jul19-en.html
Sinhala	https://www.icann.org/sites/default/files/lgr/lgr-3-sinhala-script-10jul19-en.html
Tamil	https://www.icann.org/sites/default/files/lgr/lgr-3-tamil-script-10jul19-en.html
Telugu	https://www.icann.org/sites/default/files/lgr/lgr-3-telugu-script-10jul19-en.html
Thai	https://www.icann.org/sites/default/files/lgr/lgr-3-thai-script-10jul19-en.html

For each XML file, a mechanically generated and non-normative HTML presentation, as shown in Table 2, is provided for ease of review. Any discrepancy between the XML and HTML is resolved by the XML being the primary. The HTML presentation is augmented by summary data, as well as data extracted from the Unicode Character Database [UCD], such as the character name.

Table 3. Other Files [PDF - non-normative]

Contents	File URL
Overview and Summary	This document
Repertoire Tables, non-CJK	https://www.icann.org/sites/default/files/lgr/lgr-3-non-cjk-10jul19-en.pdf

Repertoire tables are presented as non-normative PDF files that show the code points included in the repertoire for the merged LGR presented in the form of marked up tables. The presentation is similar to that used for character code charts in the Unicode Standard. The background color indicates the status of the code point:

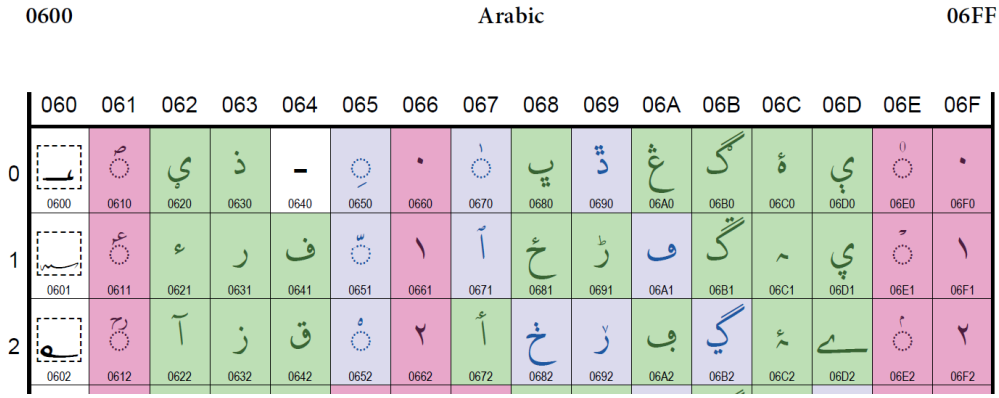

Figure 1. Sample Repertoire Table

- Green: code points that are part of the LGR, including all members of code point sequences.[3]
- White: code points that are not PVALID in IDNA 2008 [RFC5892][IDNAREG].
- Pink: code points that are ***excluded*** from the Root Zone in a generic fashion (digits, hyphen), or by being excluded from the Maximal Starting Repertoire [MSR-4].
- Lavender: code points not included in the LGR as result of decisions by the Generation Panels during the development of the LGR.

Unicode blocks that contain no repertoire of the LGR are suppressed.

2 Process of Integration

2.1 Overview

The process for developing the Root Zone LGR consists of two stages, whereby a series of community-based Generation Panels creates and submits for public review a set of Proposed LGRs for their

[3] If an LGR defines a code point sequence containing a certain code point, but not the code point by itself, a valid label may contain that code point only as part of the given sequence. However, for the purpose of these repertoire tables, such code points are shown without any distinction.

respective scripts. A separate expert panel, the Integration Panel, has the task to select from the submitted LGRs those ready for integration and to assemble them into a version of the Root Zone LGR.

The [Procedure] assumes that each Generation Panel is best situated to make the selection of code points and variants specific to its script and to propose a disposition for them in the proposed LGR. In general, it is expected that Generation Panels will propose to include only a subset of code points that are in scope for their respective scripts. Generation Panels are expected to provide an adequate rationale including references for each code point included. See also [Guidelines].

The Integration Panel is tasked to evaluate the submitted LGR proposals in light of the Principles laid out in the [Procedure].

The review of LGR proposals undertaken by the Integration panel combines mechanical review steps with qualitative review in light of a set of principles as described in Section B.4 in [Procedure].

Mechanical review steps include verifying that the proposed LGR

- is within the MSR
- is within the scope (script)
- is symmetric and transitive (with respect to variants)
- contains all default WLE rules and actions
- contains the required files
- meets the syntax requirements

The qualitative review includes evaluation of the proposed LGR against these principles set out in Section A.3.6 in [Procedure] and [IABCP]:

Least Astonishment Principle: A Code Point in the Zone Repertoire should not present recognition difficulties to the zone's intended user population and should not lend itself to malicious use.

Contextual Safety Principle: A code point in the Zone Repertoire or any of its Variants that present unacceptable risks of being used in malicious ways should not be permitted.

Simplicity Principle: Overly complex rules are to be avoided, in favor of rules easily understood by users with only some background. In particular, in the root, rules should not require deep familiarity with a particular script or language.

Predictability Principle: People with reasonable knowledge of the topic should by and large reach the same conclusions about which code points should be included.

Stability Principle: Once a code point is permitted, it is almost impossible to stop permitting it: the act of permitting a code point cannot be undone. This is particularly true once a label containing this code point has been registered.

The following principles are normally satisfied implicitly, whether by the way the overall process is organized (by inclusion) or by the way the [MSR-4] defined the boundaries for LGRs. For the inclusion

principle, in particular, the IP review checks whether all included code points are justified individually or by being part of a fixed set and documented as such.

Inclusion Principle: The zone repertoire is built up by specific inclusion; the default status for any code point is that it is excluded.

Letter Principle: Only Assigned Code Points normally used to write words should be permitted. Assigned Code Points normally used for both words and other purposes should not be permitted.

Longevity Principle: A Code Point in the Zone Repertoire should have stable properties across multiple versions of Unicode.[4]

The last principle is an overarching one that applies not only to code points, but also variants and other features of the LGR, and finally to the entire review and integration process. If there are doubts, it is best to withhold approval, rejecting or deferring a proposal until the doubt can be removed. From the Conservatism Principle also follows the prescription in [Procedure] to minimize allocatable variants and to maximize (within reason) the blocked variants.

Conservatism Principle: Any doubt should be resolved in favor of exclusion of a code point rather than inclusion.

Proposed variants are further evaluated as to whether they follow the guidelines in [RFC8228] and result in variant label sets that are well behaved, particularly with respect to index label generation (see Section 5.5 “Index Label Calculation”).

For more details on the review carried out for specific proposals, see Section 2.3.

2.2 Proposals Submitted

An integrated LGR starts from proposals for script-based LGRs. At the outset of the work on the current version of the Root Zone LGR, the following proposals had been submitted by the respective Generation Panels:

[4] Generally, that implies that code points from more recent versions of Unicode may require more stringent justification for inclusion.

Table 4. Script-Based LGR Proposals for the Root Zone

Script	Status	Files Submitted
Arabic	*in LGR-1*	arabic-lgr-proposal-18nov15-en.pdf
LGR Specification		proposed-arabic-lgr-18nov15-en.xml
Test Labels		arabic-labels-18nov15-en.txt
Armenian	*deferred*	armenian-lgr-proposal-05nov15-en.pdf
LGR Specification		proposed-armenian-lgr-05nov15-en.xml
Test Labels		armenian-test-labels-05nov15-en.txt
Cyrillic	*deferred*	proposal-cyrillic-lgr-03apr18-en.pdf
LGR Specification		proposal-cyrillic-lgr-03apr18-en.xml
Test Labels		cyrillic-test-labels-03apr18-en.txt
Devanagari	*in LGR-3*	proposal-devanagari-lgr-22apr19-en.pdf
LGR Specification		proposal-devanagari-lgr-22apr19-en.xml
Test Labels		devanagari-test-labels-22apr19-en.txt
Ethiopic	*in LGR-2*	proposal-ethiopic-lgr-17may17-en.pdf
LGR Specification		proposal-ethiopic-lgr-17may17-en.xml
Test Labels		ethiopic-test-labels-17may17-en.txt
Georgian	*in LGR-2*	proposal-georgian-lgr-24nov16-en.pdf
LGR Specification		proposal-georgian-lgr-15sep16-en.xml
Test Labels		georgian-test-labels-15sep16-en.txt
Gujarati	*in LGR-3*	proposal-gujarati-lgr-06mar19-en.pdf
LGR Specification		proposal-gujarati-lgr-06mar19-en.xml
Test Labels		gujarati-test-labels-06mar19-en.txt
Gurmukhi	*in LGR-3*	proposal-gurmukhi-lgr-22apr19-en.pdf
LGR Specification		proposal-gurmukhi-lgr-22apr19-en.xml
Test Labels		gurmukhi-test-labels-22apr19-en.txt
Hebrew	*in LGR-3*	proposal-hebrew-lgr-24apr19-en.pdf
LGR Specification		proposal-hebrew-lgr-24apr19-en.xml
Test Labels		hebrew-test-labels-24apr19-en.txt
Kannada	*in LGR-3*	proposal-kannada-lgr-06mar19-en.pdf
LGR Specification		proposal-kannada-lgr-06mar19-en.xml
Test Labels		kannada-test-labels-06mar19-en.txt
Khmer	*in LGR-2*	proposal-khmer-lgr-15aug16-en.pdf
LGR Specification		proposal-khmer-lgr-15aug16-en.xml
Test Labels		khmer-test-labels-15aug16-en.txt
Lao	*in LGR-2*	proposal-lao-lgr-31jan17-en.pdf
LGR Specification		proposal-lao-lgr-31jan17-en.xml
Test Labels		lao-test-labels-31jan17-en.txt
Malayalam	*in LGR-3*	proposal-malayalam-lgr-22apr19-en.pdf
LGR Specification		proposal-malayalam-lgr-22apr19-en.xml
Test Labels		malayalam-test-labels-22apr19-en.txt
Oriya	*in LGR-3*	proposal-oriya-lgr-06mar19-en.pdf
LGR Specification		proposal-oriya-lgr-06mar19-en.xml
Test Labels		oriya-test-labels-06mar19-en.txt

Sinhala	*in LGR-3*	proposal-sinhala-lgr-22apr19-en.pdf
LGR Specification		proposal-sinhala-lgr-22apr19-en.xml
Test Labels		sinhala-test-labels-22apr19-en.txt
Tamil	*in LGR-3*	proposal-tamil-lgr-06mar19-en.pdf
LGR Specification		proposal-tamil-lgr-06mar19-en.xml
Test Labels		tamil-test-labels-06mar19-en.txt
Telugu	*in LGR-3*	proposal-telugu-lgr-07jun19-en.pdf
LGR Specification		proposal-telugu-lgr-07jun19-en.xml
Test Labels		telugu-test-labels-07jun19-en.txt
Thai	*in LGR-2*	proposal-thai-lgr-25may17-en.pdf
LGR Specification		proposal-thai-lgr-25may17-en.xml
Test Labels		thai-test-labels-25may17-en.txt

The Integration Panel reviewed proposals submitted since the previous version of the LGR and determined whether they could be integrated into the current version of the LGR.

2.3 Review of Proposals

2.3.1 General Notes on the Proposal Review

After a thorough review, the Integration Panel was unanimous in accepting the following LGRs for Devanagari, Gujarati, Gurmukhi, Hebrew, Kannada, Malayalam, Oriya, Sinhala, Tamil and Telugu for integration into LGR-3. The Integration Panel unanimously continued the deferral of the proposed LGR for Armenian, because its interaction with other scripts cannot be fully evaluated at this time; the IP also decided to defer the proposal for Cyrillic on the same grounds. These proposals are not rejected, but deferred for review in the context of a future LGR.

The Ethiopic, Georgian, Khmer, Lao and Thai LGR had been reviewed and approved for integration into LGR-2, while the Arabic LGR had been reviewed and approved for integration into LGR-1. These LGRs continue to be integrated in LGR-3.

As result of the review of proposals submitted, the contents of LGR-3 are defined by 16 script-specific LGRs listed in Table 5 above as accepted or retained from earlier versions of the LGR, as well as by the default WLE rules and actions defined by the Integration Panel (IP) as part of the [MSR-4]. (See Section 3 for a summary of the contents of the Root Zone LGR).

The following subsections provide details on the review and disposition of specific proposals for each script. Please note:

(a) Details on the review of proposals from any previous edition of the LGR are not repeated here.
(b) The summary of the reviews of scripts included for the first time in this edition of the LGR each cover the following points:
    - Overview,
    - Highlight of particular issues encountered,
    - Scope of mechanical testing of LGR proposal,

- Scope of label testing,
- Potential for collisions with code points in any other script, and
- Disposition.

2.3.2 Arabic LGR Proposal Review

For information on the original review of [Proposal-Arabic], see Section 2.3.2 of [LGR-1].

The Arabic Script LGR has been part of the Root Zone LGR since [LGR-1]. Being upwardly compatible, the current version continues to include this script LGR unchanged from [LGR-1], except for minor editorial adjustments.

2.3.3 Armenian LGR Proposal Review

For information on the original review of [Proposal-Armenian], see Section 2.3.1 of [LGR-1].

While the Armenian LGR proposal was successfully submitted and passed mechanical and other review, the IP continues in the conclusion, that the script should be treated as being related to other scripts in the sense of Section 3.2 of MSR-4. Consequently, the IP chose to continue to defer the script until its interactions with the related scripts are well-enough understood to cause no risk of future incompatibilities.

2.3.4 Cyrillic LGR Proposal Review

The Integration Panel worked with the Cyrillic Generation Panel [Cyrillic GP] during the development of [Proposal-Cyrillic] to ensure that it would meet the Integration Panel's understanding of the [IABCP] principles and other prescriptions found in [Procedure].

In particular, this included the rejection of 02BC MODIFIER LETTER APOSTROPHE on the basis of grave security concerns raised by the Internet Architecture Board (IAB) (See [IABCP] and [IAB-Comment]). It further included reviewing which of the minority languages required additions to the repertoire and which of the related scripts should be evaluated for cross-script variants, as well as the basis on which to define these variants.

A separate mechanical review of the proposal has verified that the specification of the repertoire in the XML is valid and in accordance with [Proposal-Cyrillic]; that review further confirmed, by evaluating the supplied test labels, that the result of applying the LGR adequately reflects the understanding that went into its design.

The IP reviewed the proposed cross-script variant relation for U+04CF PALOCHKA. The typography for this code point is rather variable across fonts; a homoglyph relation exists most commonly with "l", (lowercase L), but it can also be rendered as "small-caps i" or as "dot-less i". Because those code points are not variants of each other, the requirement for transitivity prevents adding additional cross-script variants: the resulting potential for visual confusion is hereby brought to the attention of the String Similarity Review. [Proposal-Cyrillic] lists a number of other code points with varying degrees of visual similarity.

The LGR was also reviewed against any Cyrillic ccTLDs and gTLDs existing at the time of review.

While the Cyrillic LGR proposal was successfully submitted and passed mechanical and other review, the IP concludes that the script should be treated as being related to other scripts in the sense of Section 3.2 of MSR-4. Consequently, the IP chooses to defer the script until its interactions with the related scripts are well-enough understood to cause no risk of future incompatibilities.

2.3.5 Devanagari LGR Proposal Review

The Integration Panel worked with the Neo-Brahmi Generation Panel [NeoBGP] during the development of [Proposal-Devanagari] to ensure that it would meet the Integration Panel's understanding of the [IABCP] principles and other prescriptions found in [Procedure].

Devanagari is a complex script that is part of a family of related Neo-Brahmi scripts all developed by the same generation panel. A key feature is that the users process the script at the syllable (akshar) level, while the encoding breaks these down into their constituting elements, such as consonants, vowel signs, and other marks. Sequences of code points that lead to invalid aksharas must be avoided as neither users nor display engines can reliably process them. (Note that the Unicode Standard also formally declares that some of these sequences should not be used). In the LGR, this is achieved by categorizing certain classes of code points and defining context rules for them. The IP reviewed the proposed context rules and worked with the NeoBGP to make the Devanagari LGR draft a template for the work on the other scripts under the purview of the GP.

For variants, the review focused on issues with the proposed set of in-script and cross-script variants. A substantial number of cross-script variants were proposed for Gurmukhi, reflecting the relation between these scripts. A smaller number of cross-script variants were proposed for Bengali, a script initially developed jointly, but now postponed for a future version of the LGR. The NeoB GP worked to ensure the consistency of cross-script variants with all affected scripts.

For in-script variants, the proposed variants between "Vowel + Nukta" and "Vowel without Nukta" required careful handling to avoid a seeming "recursion" in the set of variant labels. See Section 6.5 "Effective Null Variants". A number of cross script variants have the effect that when used in some labels, the full variant label set would include mixed script labels, which would be invalid. However, the constraints on valid labels ensure that the Index Label for each set is computed correctly as described in Section 5.5 "Index Label Calculation".

Some variant definitions were found to interact with the code point contexts (A → B, where both A and B are allowed in different contexts). These cases were resolved by adjusting the code point contexts where possible, otherwise by adding variant contexts for in-script variants.

Finally, the proposal contains a number of overlapped variants (A → B and AC → D, where AC is a sequence containing A). Some of these cases suggested additional variant definitions (e.g. BC -> D) or context rules. Several rounds of detailed review confirmed that index label computation remains unambiguous .

Devanagari is used for a range of languages; therefore, the review included looking at whether they all appeared to be adequately supported.

A separate mechanical review of the proposal has verified that the specification of the repertoire and WLE rules in the XML are valid and in accordance with [Proposal-Devanagari]; a mechanical evaluation of the supplied test labels confirmed that the result of applying the LGR adequately reflects the understanding that went into its design.

The LGR was also reviewed against a set of putative Devanagari labels derived from text corpora in different languages, as well as against any existing Devanagari ccTLDs and gTLDs.

Based on this review and having resolved any open issues in discussion with the NeoBGP, the IP unanimously declared the Devanagari LGR Proposal ready for integration into the Root Zone LGR as submitted.

2.3.6 Ethiopic LGR Proposal Review

For information on the original review of [Proposal-Ethiopic], see Section 2.3.4 of [LGR-2].

The Ethiopic Script LGR has been part of the Root Zone LGR since [LGR-2]. Being upwardly compatible, the current version continues to include this script LGR unchanged from [LGR-2], except for minor editorial adjustments.

2.3.7 Georgian LGR Proposal Review

For information on the original review of [Proposal-Georgian], see Section 2.3.5 of [LGR-2].

The Georgian Script LGR has been part of the Root Zone LGR since [LGR-2]. Being upwardly compatible, the current version continues to include this script LGR unchanged from [LGR-2], except for minor editorial adjustments.

2.3.8 Gujarati LGR Proposal Review

The Integration Panel worked with the Neo-Brahmi Generation Panel [NeoBGP] during the development of [Proposal-Gujarati] to ensure that it would meet the Integration Panel's understanding of the [IABCP] principles and other prescriptions found in [Procedure].

Gujarati is a complex script that is part of a family of related Neo-Brahmi scripts all developed by the same generation panel. A key feature is that the users process the script at the syllable (akshar) level, while the encoding breaks these down into their constituent elements, such as consonants, vowel signs, and other marks. Sequences of code points that lead to invalid aksharas must be avoided as neither users nor display engines can reliably process them. (Note that the Unicode Standard also formally declares that some of these sequences should not be used). In the LGR, this is achieved categorizing certain classes of code points and by adding a small number of context rules for them. The IP reviewed the proposed context rules and worked with the NeoBGP to help ensure a consistency in approach and notation.

The Gujarati LGR proposal does not contain variants.

A separate mechanical review of the proposal has verified that the specification of the repertoire and WLE rules in the XML are valid and in accordance with [Proposal-Gujarati]; a mechanical evaluation of

the supplied test labels confirmed that the result of applying the LGR adequately reflects the understanding that went into its design.

The LGR was also reviewed against a set of putative Gujarati labels derived from a text corpus, as well as against any existing Gujarati ccTLDs and gTLDs.

Based on this review and having resolved any open issues in discussion with the NeoBGP, the IP unanimously decided that the Gujarati LGR Proposal is ready for integration into the Root Zone LGR as submitted.

2.3.9 Gurmukhi LGR Proposal Review

The Integration Panel worked with the Neo-Brahmi Generation Panel [NeoBGP] during the development of [Proposal-Gurmukhi] to ensure that it would meet the Integration Panel's understanding of the [IABCP] principles and other prescriptions found in [Procedure].

Gurmukhi is a complex script that is part of a family of related Neo-Brahmi scripts all developed by the same generation panel. A key feature is that the users process the script at the syllable (akshar) level, while the encoding breaks these down into their constituting elements, such as consonants, vowel signs, and other marks. Sequences of code points that lead to invalid aksharas must be avoided as neither users nor display engines can reliably process them. (Note that the Unicode Standard also formally declares that some of these sequences should not be used). In the LGR, this is achieved by categorizing certain classes of code points and adding context rules for them. The IP reviewed the proposed context rules and worked with the NeoBGP to help ensure a consistency in approach and notation.

The LGR defines a number of cross-script variants with Devanagari and Bengali. The NeoBGP ensured that the set of variants was mutually consistent.

A separate mechanical review of the proposal has verified that the specification of the repertoire and WLE rules in the XML are valid and in accordance with [Proposal-Gurmukhi]; a mechanical evaluation of the supplied test labels confirmed that the result of applying the LGR adequately reflects the understanding that went into its design.

The LGR was also reviewed against a set of putative Gurmukhi labels derived from a text corpus, as well as against any existing Gurmukhi ccTLDs and gTLDs.

Based on this review and having resolved any open issues in discussion with the NeoBGP, the IP unanimously decided that the Gurmukhi LGR Proposal is ready for integration into the Root Zone LGR as submitted.

2.3.10 Hebrew LGR Proposal Review

The Integration Panel worked with the Hebrew Generation Panel [Hebrew GP] during the development of [Proposal-Hebrew] to ensure that it would meet the Integration Panel's understanding of the [IABCP] principles and other prescriptions found in [Procedure].

Hebrew is an alphabetic script that is written from right to left. The Hebrew GP decided from the start to aim for a very conservative repertoire, excluding all combining marks due to inconsistencies in their use and differences between language communities. There are no WLE or context rules specific to Hebrew and the five in-script variants proposed are standard final forms of five characters. All 27 code points are strong right to left letters; therefore there are no writing direction related issues inside any Hebrew label, see [RFC 5893].

Based on this review and discussion with the Hebrew GP, the IP unanimously decided that the Hebrew LGR Proposal is ready for integration into the Root Zone LGR as submitted.

2.3.11 Kannada LGR Proposal Review

The Integration Panel worked with the Neo-Brahmi Generation Panel [NeoBGP] during the development of [Proposal-Kannada] to ensure that it would meet the Integration Panel's understanding of the [IABCP] principles and other prescriptions found in [Procedure].

Kannada is a complex script that is part of a family of related Neo-Brahmi scripts all developed by the same generation panel. A key feature is that the users process the script at the syllable (akshar) level, while the encoding breaks these down into their constituting elements, such as consonants, vowel signs, and other marks. Sequences of code points that lead to invalid aksharas must be avoided as neither users nor display engines can reliably process them. (Note that the Unicode Standard also formally declares that some of these sequences should not be used). In the LGR, this is achieved by categorizing certain classes of code points and adding context rules them. The IP reviewed the proposed context rules and worked with the NeoBGP to help ensure a consistency in approach and notation.

The LGR defines a substantial number of cross-script variants with Telugu. The NeoBGP ensured the internal consistency of variants between Kannada and Telugu scripts. As result of review with the Sinhala GP, a number of cross-script confusables with Sinhala were not proposed as variants.

A separate mechanical review of the proposal has verified that the specification of the repertoire and WLE rules in the XML are valid and in accordance with [Proposal-Kannada]; a mechanical evaluation of the supplied test labels confirmed that the result of applying the LGR adequately reflects the understanding that went into its design.

The LGR was also reviewed against a set of putative Kannada labels derived from a text corpus, as well as against any existing Kannada ccTLDs and gTLDs.

Based on this review and having resolved any open issues in discussion with the NeoBGP, the IP unanimously decided that the Kannada LGR Proposal is ready for integration into the Root Zone LGR as submitted.

2.3.12 Khmer LGR Proposal Review

For information on the original review of [Proposal-Khmer], see Section 2.3.6 of [LGR-2].

The Khmer Script LGR has been part of the Root Zone LGR since [LGR-2]. Being upwardly compatible, the current version continues to include this script LGR unchanged from [LGR-2], except for minor editorial adjustments.

2.3.13 Lao LGR Proposal Review

For information on the original review of [Proposal-Lao], see Section 2.3.7 of [LGR-2].

The Lao Script LGR has been part of the Root Zone LGR since [LGR-2]. Being upwardly compatible, the current version continues to include this script LGR unchanged from [LGR-2], except for minor editorial adjustments[5].

2.3.14 Malayalam LGR Proposal Review

The Integration Panel worked with the Neo-Brahmi Generation Panel [NeoBGP] during the development of [Proposal-Malayalam] to ensure that it would meet the Integration Panel's understanding of the [IABCP] principles and other prescriptions found in [Procedure].

Malayalam is a complex script that is part of a family of related Neo-Brahmi scripts all developed by the same generation panel. A key feature is that the users process the script at the syllable (akshar) level, while the encoding breaks these down into their constituting elements, such as consonants, vowel signs, and other marks. Sequences of code points that lead to invalid aksharas must be avoided as neither users nor display engines can reliably process them. (Note that the Unicode Standard also formally declares that some of these sequences should not be used). In the LGR, this is achieved by categorizing certain classes of code points and adding context rules for them. The IP reviewed the proposed context rules and worked with the NeoBGP to help ensure a consistency in approach and notation. As result of the review, it was possible to reduce the complexity of the proposed rules.

The LGR defines a number of cross-script variants with Tamil and a smaller number of variants with Oriya. The NeoBGP ensured the internal consistency of variants among these Neo-Brahmi scripts. Review discussions with the Sinhala GP determined that there are no candidates for cross-script variants with Sinhala. Two Malayalam sequences are defined as in-script variants. As one of the sequences has a context rule, for consistency the variants required a context rule as part of their variant definition.

The relatively recent addition of direct encoding for Malayalam chillu characters has led to widespread duplication in encoding as the previously existing legacy sequences using ZWJ are still widely found in data, for example, affecting the test data files. As ZWJ is not permissible in the Root Zone, these legacy sequences are automatically excluded without any explicit action. (A whole-label rule and action prevent chillu characters from beginning a label.)

Inserting U+0D4D MALAYALAM SIGN VIRAMA between two adjacent instances of U+0D33 MALAYALAM LETTER LLA results in conjunct glyph distinguished mainly by subtle kerning, which as seen as a security risk. A naïve implementation would have made the VIRAMA an effective null variant (see Section 6.4 "Effective Null Variants"), with all its attendant problems. Instead, the proposal enumerates required

[5] Editorial changes for Lao include the correction of some of the comments on code points in the XML; in the original proposal, there was a discrepancy between the XML and the supporting document.

combinations of U+0D33 with U+0D4D and uses context rules and variant context rules to ensure that the resulting variant label sets are well behaved, particularly for index label computation (see Section 5.5 “Index Label Calculation”).

A similar issue for U+0D31 MALAYALAM LETTER RRA was addressed in the same manner, with the additional complication of existing sequences containing U+0D31. This required additional context rules, but also some work in optimizing the proposed rules.

A separate mechanical review of the proposal has verified that the specification of the repertoire and WLE rules in the XML are valid and in accordance with [Proposal-Malayalam]; a mechanical evaluation of the supplied test labels confirmed that the result of applying the LGR adequately reflects the understanding that went into its design.

The LGR was also reviewed against a set of putative Malayalam labels derived from a text corpus, as well as against any existing Malayalam ccTLDs and gTLDs.

Based on this review and having resolved any open issues in discussion with the NeoBGP, the IP unanimously decided that the Malayalam LGR Proposal is ready for integration into the Root Zone LGR as submitted.

2.3.15 Oriya LGR Proposal Review

The Integration Panel worked with the Neo-Brahmi Generation Panel [NeoBGP] during the development of [Proposal-Oriya] to ensure that it would meet the Integration Panel’s understanding of the [IABCP] principles and other prescriptions found in [Procedure].

Oriya is a complex script that is part of a family of related Neo-Brahmi scripts all developed by the same generation panel. A key feature is that the users process the script at the syllable (akshar) level, while the encoding breaks these down into their constituting elements, such as consonants, vowel signs, and other marks. Sequences of code points that lead to invalid aksharas must be avoided as neither users nor display engines can reliably process them. (Note that the Unicode Standard also formally declares that some of these sequences should not be used). In the LGR, this is achieved by categorizing certain classes of code points and adding context rules for them. The IP reviewed the proposed context rules and worked with the NeoBGP to help ensure a consistency in approach and notation.

The LGR defines one cross-script variant with Malayalam and two cross-script variants with Myanmar, a script not yet proposed. However, the NeoBGP ensured the internal consistency of variants among Neo-Brahmi scripts and consulted with the Myanmar GP to remove the risk of incompatible changes. The IP separately reviewed whether security-relevant cross-script homoglyphs exist with other scripts already in or pending for inclusion in the Root Zone LGR.

A separate mechanical review of the proposal has verified that the specification of the repertoire and WLE rules in the XML are valid and in accordance with [Proposal-Oriya]; a mechanical evaluation of the supplied test labels confirmed that the result of applying the LGR adequately reflects the understanding that went into its design.

The LGR was also reviewed against a set of putative Oriya labels derived from a text corpus, as well as against any existing Oriya ccTLDs and gTLDs.

Based on this review and having resolved any open issues in discussion with the NeoBGP, the IP unanimously decided that the Oriya LGR Proposal is ready for integration into the Root Zone LGR as submitted.

2.3.16 Sinhala LGR Proposal Review

The Integration Panel worked with the Sinhala Generation Panel [Sinhala GP] during the development of [Proposal-Sinhala] to ensure that it would meet the Integration Panel's understanding of the [IABCP] principles and other prescriptions found in [Procedure].

Sinhala is a complex script that is related to other scripts in the region, such as the scripts that are the focus of the [NeoBGP]; the Sinhala GP worked in close consultation with the NeoBGP. A key feature of such scripts is that the users process the script at the syllable (akshar) level, while the encoding breaks these down into their constituting elements, such as consonants, vowel signs, and other marks. Sequences of code points that lead to invalid aksharas must be avoided as neither users nor display engines can reliably process them. (Note that the Unicode Standard also formally declares that some of these sequences should not be used). In the LGR, this is by categorizing certain classes of code points and adding context rules for them. The IP reviewed the proposed context rules and worked with the SinhGP to help ensure a consistency in approach and notation. As result of the review, it was possible to reduce the complexity of the proposed rules.

Some of the proposed variants were found to be *overlapped*; this required additional analysis to ensure that the computation of index labels remains consistent and variant label sets are well behaved.

After review, the LGR defines no cross-script variants.

A separate mechanical review of the proposal has verified that the specification of the repertoire and WLE rules in the XML are valid and in accordance with [Proposal-Sinhala]; a mechanical evaluation of the supplied test labels confirmed that the result of applying the LGR adequately reflects the understanding that went into its design.

The LGR was also reviewed against a set of putative Sinhala labels derived from a text corpus, as well as against any existing Sinhala ccTLDs and gTLDs.

Based on this review and having resolved any open issues in discussion with the SinhGP, the IP unanimously decided that the Sinhala LGR Proposal is ready for integration into the Root Zone LGR as submitted.

2.3.17 Tamil LGR Proposal Review

The Integration Panel worked with the Neo-Brahmi Generation Panel [NeoBGP] during the development of [Proposal-Tamil] to ensure that it would meet the Integration Panel's understanding of the [IABCP] principles and other prescriptions found in [Procedure].

Tamil is a complex script that is part of a family of related Neo-Brahmi scripts all developed by the same generation panel. A key feature is that the users process the script at the syllable (akshar) level, while the encoding breaks these down into their constituting elements, such as consonants, vowel signs, and other marks. Sequences of code points that lead to invalid aksharas must be avoided as neither users nor display engines can reliably process them. (Note that the Unicode Standard also formally declares that some of these sequences should not be used). In the LGR, this is achieved by categorizing certain classes of code points and adding context rules for them. The IP reviewed the proposed context rules and worked with the NeoBGP to help ensure a consistency in approach and notation. As result of the review, it was possible to reduce the complexity of the proposed rules.

While the Tamil LGR proposal contained some overlapped variants, no additional mitigation was required.

The LGR defines six cross-script variants with Malayalam and two in-script variants reflecting two alternate ways of writing the syllable SHRI. The NeoBGP ensured the internal consistency of variants among Neo-Brahmi scripts and included the Sinhala GP in these deliberations. The alternate ways of writing SHRI are an unavoidable consequence of recent changes in the preferred encoding. As a result, users will employ these interchangeably and no clear preference exists at this point. The GP decided to propose these as allocatable variants. During the review, the need to prevent arbitrary combinations of these in the same label became apparent and the LGR now has a 'no-mix' rule similar to those for Arabic.

A separate mechanical review of the proposal has verified that the specification of the repertoire and WLE rules in the XML are valid and in accordance with [Proposal-Tamil]; a mechanical evaluation of the supplied test labels confirmed that the result of applying the LGR adequately reflects the understanding that went into its design.

The LGR was also reviewed against a set of putative Tamil labels derived from a text corpus, as well as against any existing Tamil ccTLDs and gTLDs.

Based on this review and having resolved any open issues in discussion with the NeoBGP, the IP unanimously decided that the Tamil LGR Proposal is ready for integration into the Root Zone LGR as submitted.

2.3.18 Telugu LGR Proposal Review

The Integration Panel worked with the Neo-Brahmi Generation Panel [NeoBGP] during the development of [Proposal-Telugu] to ensure that it would meet the Integration Panel's understanding of the [IABCP] principles and other prescriptions found in [Procedure].

Telugu is a complex script that is part of a family of related Neo-Brahmi scripts all developed by the same generation panel. A key feature is that the users process the script at the syllable (akshar) level, while the encoding breaks these down into their constituting elements, such as consonants, vowel signs, and other marks. Sequences of code points that lead to invalid aksharas must be avoided as neither users nor display engines can reliably process them. (Note that the Unicode Standard also formally declares that some of these sequences should not be used). In the LGR, this is achieved by categorizing certain classes of code points and adding context rules for them. The IP reviewed the proposed context rules and worked with the NeoBGP to help ensure a consistency in approach and notation. As result of the review, it was possible to reduce the complexity of the proposed rules.

The LGR defines a substantial number of cross-script variants with Kannada. The NeoBGP ensured the internal consistency of variants between Kannada and Telugu scripts. As result of review with the Sinhala GP, a number of cross-script confusables with Sinhala were not proposed as variants.

In response to public comments, a rule restricting Halant from following nasal consonants was removed by the GP; the existence of an alternate spelling using Anusvara notwithstanding. The position is that such spelling issues are out of scope of the LGR (compare “color” vs. “colour” in English). This change brings the rules into alignment with Kannada, which, in all respects is a very closely related script.

A separate mechanical review of the proposal has verified that the specification of the repertoire and WLE rules in the XML are valid and in accordance with [Proposal-Telugu]; a mechanical evaluation of the supplied test labels confirmed that the result of applying the LGR adequately reflects the understanding that went into its design.

The LGR was also reviewed against a set of putative Telugu labels derived from a text corpus, as well as against any existing Telugu ccTLDs and gTLDs.

Based on this review and having resolved any open issues in discussion with the NeoBGP, the IP unanimously decided that the Telugu LGR Proposal is ready for integration into the Root Zone LGR as submitted.

2.3.19 Thai LGR Proposal Review

For information on the original review of [Proposal-Thai], see Section 2.3.8 of [LGR-2].

The Thai Script LGR has been part of the Root Zone LGR since [LGR-2]. Being upwardly compatible, the current version continues to include this script LGR unchanged from [LGR-2], except for minor editorial adjustments.

3 Integration and Contents of LGR-3

3.1 General Notes

After reviewing and accepting a proposed LGR, the Integration panel prepares an XML file containing an equivalent LGR as measured in terms of valid labels and variants produced, but with changes to the metadata and comments for consistency with the other elements of an integration process for the Root

Zone LGRs. Collectively, these constitute the Element LGRs. From each an annotated HTML file is created mechanically for a more human-readable presentation of the data.

Element LGRs included from earlier versions of the LGR are updated as to version number and date; minor changes to other metadata and comments for consistency are also applied.

From the Element LGRs a merged XML file is created mechanically containing the union of the repertoire and non-reflexive variant mappings and annotating each item in the repertoire and rules to mark its origin in a particular element LGR. This file constitutes the Common LGR. Because the actual type of all variant mappings is script-specific and therefore cannot be represented in a merged file, all variant mappings are set to "blocked" in the merged file (See also Section 5).

While script-specific tags, rules and classes are prefixed with a script name and individually included, all actions and default WLE rules from the Element LGRs are coalesced in the merged file. In principle, the default WLE rules and any actions are not script-specific, but in practice, they are usually triggered by ranges of code points or variant types specific to an element LGR. The IP manually reviews the result to make sure that these elements from different LGRs do not conflict. If necessary, they are restated. Finally, an annotated, human-readable presentation of the merged file is created.

The common LGR being generated by a software program from the Element LGRs, if a discrepancy were to be discovered, the way to resolve it would be to recalculate the Common LGR from the source Element LGRs and reissue a corrected version of the Common LGR.

The following sections summarize briefly the contents of particular files making up the Root Zone LGR. These files are listed in Tables 1 and 2 above. For more details and background on the organization of the LGR across files, see [Packaging].

3.1.1 Summary

The following table presents a summary of LGR-3 (as well as currently deferred LGRs) giving repertoire size, number and types of variant as well as numbers of script-specific rules and actions.

Table 5. Summary LGR contents (including deferred LGRs)

Script	Code Points[6]	Sequ.	Variant Sets	Allocatable	Blocked	Out-of-Rep	Rules	Actions
Arab	128		16	26	166		16	16
Armn	38		9		36	13		
Cyrl	86		28		74	32		
Deva	83+1	27	40		122	28	6	
Ethi	311		30		98			
Georg	33							
Gujr	65						3	
Guru	56	5	25		76	30	6	

[6] The count includes code points that are only available as part of a defined sequence.

Hebr	27		5		10			
Khmr	71	2	1		2		12	1
Knda	62		34		68	34	3	
Laoo	51	1					9	
Mlym	70	10	12		24	7	12	1
Orya	62		2		8	3	3	
Sinh	72	4	9		18		4	
Taml	48	4	9	2	16	6	2	
Telu	63		34		68	34	3	
Thai	68+1	3					6	
Merged	1289	46	145	N/A	502	(4)	72	19

Italic: deferred LGR

bold: added or updated in LGR-3

Notes: due to overlapping definitions and not counting the deferred LGRs, the numbers for the Merged LGR total are not equal to the sum of the values for the element LGRs in the same column. Rules and actions reflect the number of script specific named rules and any associated actions in the XML files.

3.2 Merged LGR (Common)

3.2.1 Repertoire

The repertoire of the merged Root Zone LGR is the cumulative repertoire of all the Element LGRs that have been integrated into this version. Those repertoires, in turn were developed based on [MSR-4], which is a subset the PVALID code points in IDNA2008, which at the time were a subset of Unicode 6.3 [Unicode 6.3]. The MSR excludes code points used for historical or special purposes only, or those used in languages that did not meet the criteria for stable and modern usage as outlined in [MSR-4].

As appropriate for the Root Zone LGR, the repertoire includes neither digits nor the HYPHEN-MINUS.

The merged repertoire contains all sequences defined by the Element LGRs. If any code point that is a member of a sequence is not also listed by itself in an Element LGR, it will not be defined by itself in the merged LGR. Root Zone labels may contain that code point, but only as part of a defined sequence.

3.2.2 Variants

The variant mappings in the merged LGR are the union of the non-reflexive variant mappings from all the Element LGRs that have been integrated into this version of the Root Zone LGR. Unlike the Element LGRs, the merged LGR does not contain code points with reflexive mappings of “out-of-repertoire-var”, nor any variant mappings to them.

Because the dispositions of variant labels, for example as "allocatable", are specific to each script, they cannot be expressed in the script-neutral context of this integrated LGR. Instead, in the merged LGR, all variant mappings are given the type "blocked”. (This allows the use of the Common LGR in checking for conflicts between labels as described in Section 5.4.)

3.2.3 Character Classes

The character classes in the merged LGR are the union of the character classes from all the Element LGRs that have been integrated into this version of the Root Zone LGR. Many character classes are derived in turn from tag values associated with code points in the repertoire. These tag values have also been merged. To avoid duplications, the names of all tags and character classes in the merged LGR are prefixed by the four-letter Unicode script identifier identifying the Element LGR from which they were merged.

3.2.4 Whole-Label Evaluations (WLE) Rules

The merged LGR includes the cumulative set of Whole-Label Evaluation rules and actions for all Element LGRs that have been integrated into this version. WLE rules include both context rules and whole-label rules. The purpose of WLE rules and actions for the Root Zone LGR is to allow automatic exclusion of labels that present particular challenges in display and processing, such as a label leading off with a combining mark, because that mark would tend to combine visually with the code point in front of it. Based on [Procedure] the Root Zone LGR has a single set of WLE rules that is common to all scripts. In practice, most rules are written to be specific to only the code points encountered in labels of a given script, so that the rules do not interact with each other. Each Element LGR only contains rules that are specified to it (as well as any default rules) while the IP has reviewed and made sure that the combined rules in the merged LGR do not give rise to conflicts.

To make the merged set of rules easier to follow and to avoid unintentional naming conflicts, the names of any context or whole-label rules defined by an Element LGR have been prefixed by the four-letter Unicode script identifier for that LGR before being merged into the Common LGR. The same has been done for tags and character classes. Finally, all repertoire code points have been tagged with the Unicode short identifiers for each script they are used with[7], prefixed with “sc:”(see [UAX24]).

[MSR-4] defines a number of default rules and actions. These are present in all Element LGRs and in the Common LGR. They have been annotated in the Common LGR with the prefix “Common-“.

Actions are merged, preserving their relative order of precedence from the Element LGR.

For additional details on the merged LGR, see Section 5.3 below.

The following subsections give a brief summary of the contents of each of the Element LGRs contained in this version of the Root Zone LGR. The full definition of the element LGRs is provided in files listed in Tables 1 and 2 above. (In addition, the files in Table 3 above provide a visual summary of the contents of the repertoire of the Root Zone LGR).

[7] Code points used with more than one script as identified by the Unicode Script Extension property are tagged with a list of script identifiers; all others have a single script identifier. For the Root Zone LGR, script identifiers not associated with the Root Zone are suppressed.

3.3 Arabic Element LGR

3.3.1 Repertoire for Arabic

The repertoire for the Arabic Element LGR is described in Section 3.2 in [Proposal-Arabic] by the Task Force for Arabic IDNs [TF-AIDN]. It includes only the 128 code points used by languages that are actively written in the Arabic script. It excludes code points for which TF-AIDN was unable to find sufficient evidence of use (see Appendix F in [Proposal-Arabic]).

The Arabic Element LGR does not include combining marks or code point sequences. All combining marks have been excluded for these reasons:

- First, they can significantly overproduce and would require additional rules to constrain them effectively, complicating the design.
- Second, even where they are required for some languages, they are optional for others.
- Third, this also circumvents the issue regarding duplication between some precomposed code points and combining sequences raised by [IAB].

As part of the Root Zone, the element LGR includes neither digits nor the HYPHEN-MINUS. While the script uses ZWNJ, for example in Persian, this code point is prohibited in the Root Zone. Arabic is written Right-To-Left.

For further details, see Section 3.2 "Code point repertoire included", in [Proposal-Arabic].

The Arabic LGR was first included in [LGR-1].

3.3.2 Variants for Arabic

The Arabic Element LGR includes "blocked" and "allocatable" variants, assigned according to Section 4 "Final recommendation of variants for Top Level Domains (TLDs)" in [Proposal-Arabic]. These recommendations balance the desire to minimize the number of possible allocatable variants with the need to keep the definition of variants simple.

3.3.3 Whole-Label Evaluation Rules for Arabic

The Arabic Element LGR includes Whole-Label Evaluation rules specific to the Arabic script. See Section 5 "Whole-Label Evaluation (WLE) rules", in [Proposal-Arabic]. As specified, these rules serve to prevent the mixing of two variants of the same code point within the same label. This has the effect of reducing overproduction of allocatable variant labels. See also the comments given for each rule or action.

3.3.4 Default Whole-Label Evaluation Rules

The Arabic Element LGR includes the set of required default WLE rules and actions applicable to the Root Zone and defined in [MSR-4].

3.4 Devanagari Element LGR

3.4.1 Repertoire for Devanagari

The repertoire for the Devanagari Element LGR is described in Section 5 of [Proposal-Devanagari]. It includes the 84 code points used to write modern languages in widespread common use and commonly written in the Devanagari script. Also included are 27 sequences; one code point, U+0931, only occurs as part of two sequences; thus it is not listed by itself as a member of the repertoire.

The Devanagari script is consonants and independent vowels as base letters and combining marks for dependent vowels and other signs. A special combining mark, U+094D DEVANAGARI SIGN VIRAMA, removes the inherent vowel of the preceding consonant and participates in the formation of conjuncts.

As part of the Root Zone, the element LGR includes neither digits nor the HYPHEN-MINUS. While the script formerly made use of ZWJ and may make some use of ZWNJ, these code points are prohibited in the Root Zone.

3.4.2 Variants for Devanagari

As described in Section 6 of [Proposal-Devanagari], the element LGR includes a large number of cross-script variants with related scripts, principally Gurmukhi; all are of type “blocked”. In addition, a number of in-script variants are defined; all are of type “blocked”. Some of the in-script variants involving Nukta represent “effective null variants” (See Section 6.4); for these and the “overlapped” variants involving Candrabindu there is associated context —they are only defined for labels satisfying that context (see Section 6.1.2. of [Proposal-Devanagari]). Many of the sequences in the LGR are defined because they have in-script or cross-script variants. Context rules for these sequences, in conjunction with context rules on the variants ensure that the variant label set is well behaved (see [RFC8228]).

3.4.3 Whole-Label Evaluation Rules for Devanagari

The Devanagari script uses combining marks for dependent vowels and other signs. These code points cannot occur in all contexts and the Devanagari Element LGR implements the context rules defined in Section 7 of [Proposal-Devanagari] to prevent their occurrence in contexts that could give rise to security risks.

3.4.4 Default Whole-Label Evaluation Rules

The Devanagari Element LGR includes the set of required default WLE rules and actions applicable to the Root Zone and defined in [MSR-4].

3.5 Ethiopic Element LGR

3.5.1 Repertoire for Ethiopic

The repertoire for the Ethiopic Element LGR is described in Section 5 of [Proposal-Ethiopic]. It includes only the 311 code points from the Ethiopic script needed to write languages commonly using the Ethiopic script.

The element LGR does not include combining marks or sequences.

As part of the Root Zone, the element LGR includes neither digits nor the HYPHEN-MINUS.

3.5.2 Variants for Ethiopic

As described in Section 6 of [Proposal-Ethiopic], the element LGR includes a number of variants for code points that are homophones in Amharic.

3.5.3 Whole-Label Evaluation Rules for Ethiopic

The element LGR includes no script-specific WLE rules.

3.5.4 Default Whole-Label Evaluation Rules

The Ethiopic Element LGR includes the set of required default WLE rules and actions applicable to the Root Zone and defined in [MSR-4].

3.6 Georgian Element LGR

3.6.1 Repertoire for Georgian

The repertoire for the Georgian Element LGR is described in Section 5 of [Proposal-Georgian]. It includes only the 33 code points from the Mkhedruli alphabet that are needed to write modern Georgian, a set also sufficient to write the other languages widely used and commonly written with the Georgian script.

The element LGR does not include combining marks or sequences.

As part of the Root Zone, the element LGR includes neither digits nor the HYPHEN-MINUS.

3.6.2 Variants for Georgian

The element LGR includes no variants.

3.6.3 Whole-Label Evaluation Rules for Georgian

The element LGR includes no script-specific WLE rules.

3.6.4 Default Whole-Label Evaluation Rules

The Georgian Element LGR includes the set of required default WLE rules and actions applicable to the Root Zone and defined in [MSR-4].

3.7 Gujarati Element LGR

3.7.1 Repertoire for Gujarati

The repertoire for the Gujarati Element LGR is described in Section 5 of [Proposal-Gujarati]. It includes only the 65 code points used to write modern languages in widespread common use and commonly written in the Gujarati script.

The Gujarati script is a complex script that uses consonants and independent vowels as base letters and combining marks for dependent vowels and other signs. A special combining mark, U+0ACD GUJARATI SIGN VIRAMA, removes the inherent vowel of the preceding consonant and participates in the formation of conjuncts.

As part of the Root Zone, the element LGR includes neither digits nor the HYPHEN-MINUS. While the script may use ZWJ and ZWNJ in certain cases, these code points are prohibited in the Root Zone.

3.7.2 Variants for Gujarati

The element LGR does not define any variants.

3.7.3 Whole-Label Evaluation Rules for Gujarati

The Gujarati script uses combining marks for dependent vowels and other signs. These code points cannot occur in all contexts and the Gujarati Element LGR implements the context rules defined in Section 7 of [Proposal-Gujarati] to prevent their occurrence in contexts that could give rise to security risks.

3.7.4 Default Whole-Label Evaluation Rules

The Gujarati Element LGR includes the set of required default WLE rules and actions applicable to the Root Zone and defined in [MSR-4].

3.8 Gurmukhi Element LGR

3.8.1 Repertoire for Gurmukhi

The repertoire for the Gurmukhi Element LGR is described in Section 5 of [Proposal-Gurmukhi]. It includes only the 56 code points used to write modern languages in widespread common use and commonly written in the Gurmukhi script.

The Gurmukhi script is a complex script that uses consonants and independent vowels as base letters and combining marks for dependent vowels and other signs. A special combining mark, U+0A4D VIRAMA, removes the inherent vowel of the preceding consonant and participates in the formation of conjuncts.

As part of the Root Zone, the element LGR includes neither digits nor the HYPHEN-MINUS.

3.8.2 Variants for Gurmukhi

As described in Section 6 of [Proposal-Gurmukhi], the element LGR includes a large number of cross-script variants with related scripts, principally Devanagari; all are of type “blocked”. In some case, the variants are to sequences in Devanagari. In addition two vowel diacritics are in-script variants, also of type “blocked”.

3.8.3 Whole-Label Evaluation Rules for Gurmukhi

The Gurmukhi script uses combining marks for dependent vowels and other signs. These code points cannot occur in all contexts and the Gurmukhi Element LGR implements the context rules defined in Section 7 of [Proposal-Gurmukhi] to prevent their occurrence in contexts that could give rise to security risks.

3.8.4 Default Whole-Label Evaluation Rules

The Gurmukhi Element LGR includes the set of required default WLE rules and actions applicable to the Root Zone and defined in [MSR-4].

3.9 Hebrew Element LGR

3.9.1 Repertoire for Hebrew

The repertoire for the Hebrew Element LGR is described in Section 5 of [Proposal-Hebrew]. It includes 27 unique code points, 5 of which are variants (final forms) of 5 others.

The repertoire supports the Hebrew and Yiddish languages; all combining marks have been excluded because of the variability of their use and the security concerns that they would raise.

As part of the Root Zone, the element LGR includes neither digits nor the HYPHEN-MINUS. Hebrew is written Right-to-Left.

3.9.2 Variants for Hebrew

As described in Section 6 of [Proposal-Hebrew] there are five code points that are final forms of other letters. These resulted in five pairs of blocked variants. There are no cross-script variants.

3.9.3 Whole-Label Evaluation Rules for Hebrew

The element LGR includes no script-specific WLE rules.

3.9.4 Defaults Whole-Label Evaluation Rules

The Hebrew Element LGR includes the set of required default WLE rules and actions applicable to the Root Zone and defined in [MSR-4].

3.10 Kannada Element LGR

3.10.1 Repertoire for Kannada

The repertoire for the Kannada Element LGR is described in Section 5 of [Proposal-Kannada]. It includes only the 62 code points used to write modern languages in widespread common use and commonly written in the Kannada script.

The Kannada script is a complex script that uses consonants and independent vowels as base letters and combining marks for dependent vowels and other signs. A special combining mark, U+0CCD KANNADA SIGN VIRAMA, removes the inherent vowel of the preceding consonant and participates in the formation of conjuncts.

As part of the Root Zone, the element LGR includes neither digits nor the HYPHEN-MINUS. While the script uses both ZWJ and ZWNJ, these code points are prohibited in the Root Zone.

3.10.2 Variants for Kannada

As described in Section 6 of [Proposal-Gurmukhi], the element LGR includes 34 cross-script variants with Telugu, a closely related script; all of these are of type “blocked”.

3.10.3 Whole-Label Evaluation Rules for Kannada

The Kannada script uses combining marks for dependent vowels and other signs. These code points cannot occur in all contexts and the Kannada Element LGR implements the context rules defined in

Section 7 of [Proposal-Kannada] to prevent their occurrence in contexts that could give rise to security risks.

3.10.4 Default Whole-Label Evaluation Rules

The Kannada Element LGR includes the set of required default WLE rules and actions applicable to the Root Zone and defined in [MSR-4].

3.11 Khmer Element LGR

3.11.1 Repertoire for Khmer

The repertoire for the Khmer Element LGR is described in Section 5 of [Proposal-Khmer]. It includes only the 71 code points used to write modern languages in widespread common use and commonly written in the Khmer script.

The Khmer script is a complex script that uses consonants and independent vowels as base letters and combining marks for dependent vowels and other signs. A special combining mark, U+17D2 KHMER SIGN COENG, forms sequences with following consonants that are to be rendered as subscripted form. The Khmer Repertoire explicitly lists two of these subjoined consonant sequences because of the variant relationship established between them.

As part of the Root Zone, the element LGR includes neither digits nor the HYPHEN-MINUS.

3.11.2 Variants for Khmer

The Khmer Element LGR includes two sequences for subjoined consonants that are “blocked” variants of each other due to identical appearance. When not subjoined, these consonants are not variants of each other. See Section 6 in [Proposal-Khmer].

3.11.3 Whole-Label Evaluation Rules for Khmer

The Khmer script uses combining marks for dependent vowels and other signs. These code points cannot occur in all contexts and the Khmer Element LGR implements the context rules defined in Section 7 of [Proposal-Khmer] to prevent their occurrence in contexts that could give rise to security risks; also defined is a whole-label rule limiting the number of adjacent subjoined consonant sequences.

3.11.4 Default Whole-Label Evaluation Rules

The Khmer Element LGR includes the set of required default WLE rules and actions applicable to the Root Zone and defined in [MSR-4].

3.12 Lao Element LGR

3.12.1 Repertoire for Lao

The repertoire for the Lao Element LGR is described in Section 5 of [Proposal-Lao]. It includes only the 51 code points used to write modern languages in widespread common use and commonly written in the Lao script.

The Lao script is a complex script using consonants as base letters and combining marks for vowels and other signs. The Lao Repertoire explicitly lists one sequence of vowel marks because it occurs in a specific context.

As part of the Root Zone, the element LGR includes neither digits nor the HYPHEN-MINUS.

3.12.2 Variants for Lao

The element LGR includes no variants.

3.12.3 Whole-Label Evaluations Rules for Lao

The Lao script uses combining marks for vowels, tone marks and other signs. These signs cannot occur in all contexts and the Lao Element LGR implements the context rules defined in Section 7 of [Proposal-Lao] to prevent their occurrence in contexts that could give rise to security risks; also defined is a context rule limiting the number of adjacent repetition marks at the end of the label.

To reduce complexity, the rules allow many labels that users would reject as impossible to occur in the context of writing Lao, but that represent no security risk. In contrast, a small number of words cannot be represented as labels under this LGR; a tradeoff deemed acceptable to the Lao GP as accommodating them would have required special cases to be added to the rules.

3.12.4 Default Whole-Label Evaluation Rules

The Lao Element LGR includes the set of required default WLE rules and actions applicable to the Root Zone and defined in [MSR-4].

3.13 Malayalam Element LGR

3.13.1 Repertoire for Malayalam

The repertoire for the Malayalam Element LGR is described in Section 5 of [Proposal-Malayalam]. It includes 70 code points and 10 sequences.

The Malayalam script is a complex script that uses consonants and independent vowels as base letters and combining marks for dependent vowels and other signs. A special combining mark, U+0D4D MALAYALAM SIGN VIRAMA, removes the inherent vowel of the preceding consonant and participates in the formation of conjuncts.

The relatively recent addition of direct encoding for chillu characters in Unicode would have created the potential of duplication with legacy sequences for these using ZWJ; however, this issue cannot arise because ZWJ is prohibited in the Root Zone. Nevertheless, these legacy sequences are still rather common in ordinary text data and may present an issue for users trying to type in a Malayalam TLD label unless implementers support suitable conversion.

As part of the Root Zone, the element LGR includes neither digits nor the HYPHEN-MINUS. While the script makes use of ZWNJ for orthographic uses and ZWJ for stylistic ones, these code points are prohibited in the Root Zone.

3.13.2 Variants for Malayalam

As described in Section 6 of [Proposal-Malayalam], the element LGR includes a number of cross-script variants principally with Tamil and Oriya; all of type "blocked". Several sets of code point sequences are near homographs of each other; they are defined as in-script variants of type "blocked". In some cases, the variants are *effective null variants* (See Section 6.4). To make the variant label sets well behaved and following the guidance in [RFC8228], both sequences and variant mappings have context rules. (See Section 6.1 of [Proposal-Malayalam]).

3.13.3 Whole-Label Evaluation Rules for Malayalam

The Malayalam script uses combining marks for dependent vowels and other signs. These code points cannot occur in all contexts and the Malayalam Element LGR implements the context rules defined in Section 7 of [Proposal-Malayalam] to prevent their occurrences in contexts that could give rise to security risks. Several sequences have been defined so as to override a context rule otherwise applicable to U+0D33 MALAYALAM LETTER LLA or U+0D31 MALAYALAM LETTER RRA; a context rule not being evaluated between code points in the same sequence. A whole label rule and associated action prevent chillu code points from starting a label.

3.13.4 Default Whole-Label Evaluation Rules

The Malayalam Element LGR includes the set of required default WLE rules and actions applicable to the Root Zone and defined in [MSR-4].

3.14 Oriya (Odia)[8] Element LGR

3.14.1 Repertoire for Oriya

The repertoire for the Oriya Element LGR is described in Section 5 of [Proposal-Oriya]. It includes only the 63 code points used to write modern languages in widespread common use and commonly written in the Oriya script, also known as Odia.

The Oriya script is a complex script that uses consonants and independent vowels as base letters and combining marks for dependent vowels and other signs. A special combining mark, U+0B4D ORIYA SIGN VIRAMA, removes the inherent vowel of the preceding consonant and participates in the formation of conjuncts.

As part of the Root Zone, the element LGR includes neither digits nor the HYPHEN-MINUS. While the script may use ZWJ and ZWNJ in certain cases, these code points are prohibited in the Root Zone.

3.14.2 Variants for Oriya

As described in Section 6 of [Proposal-Oriya], the element LGR includes a small number of cross-script variants to other scripts; all are of type "blocked".

[8] The Root Zone LGR uses the naming conventions from [ISO 15924] for script names. For general use, the name "Odia" is used for this script.

3.14.3 Whole-Label Evaluation Rules for Oriya

The Oriya script uses combining marks for dependent vowels and other signs. These code points cannot occur in all contexts and the Oriya Element LGR implements the context rules defined in Section 7 of [Proposal-Oriya] to prevent their occurrence in contexts that could give rise to security risks.

3.14.4 Default Whole-Label Evaluation Rules

The Oriya Element LGR includes the set of required default WLE rules and actions applicable to the Root Zone and defined in [MSR-4].

3.15 Sinhala Element LGR

3.15.1 Repertoire for Sinhala

The repertoire for the Sinhala Element LGR is described in Section 5 of [Proposal-Sinhala]. It includes 72 code points and 4 sequences.

The Sinhala script is a complex script that uses consonants and independent vowels as base letters and combining marks for dependent vowels and other signs. A special combining mark, U+0DCA SINHALA SIGN AL-LAKUNA, removes the inherent vowel of the preceding consonant and participates in the formation of conjuncts.

As part of the Root Zone, the element LGR includes neither digits nor the HYPHEN-MINUS. While the script prominently uses ZWJ, this code points is prohibited in the Root Zone.

3.15.2 Variants for Sinhala

As described in Section 6 of [Proposal-Sinhala], the element LGR includes no cross-script variants. Four sequences of code points are near homographs of singleton code points. In addition, several pairs of code points are very difficult to distinguish. All of these have been made in-script variants of type “blocked”.

3.15.3 Whole-Label Evaluation Rules for Sinhala

The Sinhala script uses combining marks for dependent vowels and other signs. These code points cannot occur in all contexts and the Sinhala Element LGR implements the context rules defined in Section 7 of [Proposal-Sinhala] to prevent their occurrences in contexts that could give rise to security risks.

3.15.4 Default Whole-Label Evaluation Rules

The Sinhala Element LGR includes the set of required default WLE rules and actions applicable to the Root Zone and defined in [MSR-4].

3.16 Tamil Element LGR

3.16.1 Repertoire for Tamil

The repertoire for the Tamil Element LGR is described in Section 5 of [Proposal-Tamil]. It includes 48 code points and 4 sequences.

The Tamil script is a complex script that uses consonants and independent vowels as base letters and combining marks for dependent vowels and other signs. A special combining mark, U+0BCD TAMIL SIGN VIRAMA, removes the inherent vowel of the preceding consonant and participates in the formation of conjuncts.

As part of the Root Zone, the element LGR includes neither digits nor the HYPHEN-MINUS. While the script makes limited use of ZWNJ, this code point is prohibited in the Root Zone.

3.16.2 Variants for Tamil

As described in Section 6 of [Proposal-Tamil], the element LGR includes a number of cross-script variants with the related script Malayalam; these are all of type “blocked”. Four sequences are defined as in-script variants. Two of them are “blocked” variants to single code points; the other two are alternate representations for the syllable /shri/ and are “allocatable” variants of each other. A special WLE rule prevents labels that mix the two representations.

3.16.3 Whole-Label Evaluation Rules for Tamil

The Tamil script uses combining marks for dependent vowels and other signs. These code points cannot occur in all contexts and the Tamil Element LGR implements the context rules defined in Section 7 of [Proposal-Tamil] to prevent their occurrences in contexts that could give rise to security risks. Also implemented is a whole-label rule with corresponding action to limit the possible number of allocatable variant labels for any label to two.

3.16.4 Default Whole-Label Evaluation Rules

The Tamil Element LGR includes the set of required default WLE rules and actions applicable to the Root Zone and defined in [MSR-4].

3.17 Telugu Element LGR

3.17.1 Repertoire for Telugu

The repertoire for the Telugu Element LGR is described in Section 5 of [Proposal-Telugu]. It includes only the 63 code points used to write modern languages in widespread common use and commonly written in the Telugu script.

The Telugu script is a complex script that uses consonants and independent vowels as base letters and combining marks for dependent vowels and other signs. A special combining mark, U+0C4D TELUGU SIGN VIRAMA, removes the inherent vowel of the preceding consonant and participates in the formation of conjuncts.

As part of the Root Zone, the element LGR includes neither digits nor the HYPHEN-MINUS. While the script may use ZWJ and ZWNJ in certain cases, these code points are prohibited in the Root Zone.

3.17.2 Variants for Telugu

As described in Section 6 of [Proposal-Telugu], the element LGR includes 34 cross-script variants with Kannada, a closely related script; all of these are of type “blocked”.

3.17.3 Whole-Label Evaluation Rules for Telugu

The Telugu script uses combining marks for dependent vowels and other signs. These code points cannot occur in all contexts and the Telugu Element LGR implements the context rules defined in Section 7 of [Proposal-Telugu] to prevent their occurrence in contexts that could give rise to security risks.

3.17.4 Default Whole-Label Evaluation Rules

The Telugu Element LGR includes the set of required default WLE rules and actions applicable to the Root Zone and defined in [MSR-4].

3.18 Thai Element LGR

3.18.1 Repertoire for Thai

The repertoire for the Thai Element LGR is defined in Section 5 of [Proposal-Thai]. It includes only the 69 code points used to write modern languages in widespread common use and commonly written in the Thai script.

The Thai script is a complex script using consonants as base letters and combining marks for vowels and other signs. The Thai Repertoire explicitly lists one sequence of vowel marks and two sequences of consonants because they occur in a specific context. One code point, U+0E45, only occurs as part of a sequence; thus, it is not listed by itself as a member of the repertoire.

The code point U+0E33, representing one of the Thai vowels, is DISALLOWED in IDNA 2008. In labels, this code point must be expressed as the sequence U+0E30 U+0E4D instead. This sequence is explicitly a member of the repertoire, to allow the exceptional occurrence of U+0E4D after a specific above-vowel.

As part of the Root Zone, the element LGR includes neither digits nor the HYPHEN-MINUS.

3.18.2 Variants for Thai

The Thai element LGR includes no variants.

3.18.3 Whole-Label Evaluations Rules for Thai

Thai is a complex script in which a set of code points create a character-cluster in a cell, and only a subset of all possible code point sequences would ever expected to occur. However, the WLE rules defined in Section 7 of [Proposal-Thai] are used to limit the contexts in which certain code points (including some consonants, vowels, tone and diacritics) may appear in the coded sequence. These ensure that the characters occur in the order expected (and supported) by typical rendering engines: they are not intended to enforce ‘spelling-rules’.

The whole-label evaluation rules for the Thai LGR would need to be relaxed over those in use for the Thai language to fully cover patterns that occur in other languages using the Thai script. However, that is not possible due the fact that unstable rendering for those patterns creates a security concern, where rendering presently becomes unreliable.

To use the simple generalized WLE Rules will also allow the user of other languages to be able to input a string in their language using the Thai Script without any limitation like spelling rules, while maintaining the consistent ordering expected by rendering engines.

3.18.4 Default Whole-Label Evaluation Rules

The Thai Element LGR includes the set of required default WLE rules and actions applicable to the Root Zone and defined in [MSR-4].

4 General Notes on the Root Zone LGR

4.1 Rules

Label Generation Rules (LGR) is the term used to describe the sets of code points, and the constraints on them, that are needed to generate IDNs in a particular script (e.g. Latin, Arabic, or Japanese).

Most of the information in a typical LGR takes the form of selections from a repertoire of code points defined in the Unicode Standard, further reduced by [MSR-4] in the case of the Root Zone. The “R” in LGR stands for “Rules” rather than “Repertoires”, because labels must be constructed out of permitted code points in context, including allowing sequences of code points as repertoire items. The validity of labels is determined by mechanically evaluating the LGR, and in particular, the Whole-Label Evaluation (WLE) rules, which use the wider context of a label. In addition, variant rules define what variant labels might exist and whether they are or are not available for allocation.

4.2 Scripts

In defining labels fit to be used globally in the DNS root zone, any code point is defined as belonging to a script, with some code points used with multiple scripts, as defined by the Script_Extensions property in the Unicode Character Database [UCD]. For the root zone, all code points used in a given label must normally belong to a single script; although any script supported in the LGR may be used to create a root label, and those labels can in principle be used anywhere in the internet, there cannot be a mixture of scripts represented within a single root label. Notably, for example, root zone LGRs for any script other than Latin cannot introduce US-ASCII code points into their repertoire.

The definition of script for used in the LGR process is that chosen by [ISO 15924]; for example, this definition recognizes that Japanese is written with a mixture of scripts, in this case, a mixture of Han ideographs with Kana, and provides separate script identifiers for such composite scripts.

Many scripts, such as Arabic, Cyrillic, Devanagari and Latin each support a variety of languages. As long as the code points are members of the same script, as defined by [ISO15924], code points used for

different languages can be mixed in a label; subject only to constraint on mixing that might be present in the WLE rules of the respective LGR.

4.3 Comprehensiveness and Staging

Ideally, the Root Zone LGR would be comprehensive, that is, include all scripts eligible for the root zone from its first version. With respect to the *Stability Principle* and the *Least Astonishment Principle* [IABCP] a fully comprehensive LGR would guarantee that all issues relating to the possible interaction among all scripts can be fully investigated in the development of the LGR. From a practical perspective doing so would be prohibitive because of the additional time needed to investigate certain scripts, and perhaps unnecessary for two main reasons.

First, not all scripts are related closely enough so that they affect each other from the perspective of LGR development. Second, it is not realistic to expect that Generation Panels will be formed and complete their work for all eligible scripts within the same time frame. Consequently, the [Procedure] anticipated that LGR would be rolled out in stages.

The goal for all future versions of the LGR must be to retain full backward compatibility, so that they preserve the output of any label registration against the old LGR, when applied to an updated LGR. Consequently, the IP anticipates that succeeding versions of the LGR will be strict supersets of its predecessors. It is expected that registrations that predate the initial release of an LGR covering the respective script will be allowed to remain, even if in conflict, but without becoming a binding precedent for the LGR itself. To date, there is no known instance of such a conflict.

5 Using the LGR

5.1 Element LGRs

The merged file containing the Common LGR and the per-script Element LGRs serve different purposes. At the time of registration, the applicant selects the script in the context of which the label is to be applied. That selection determines which element LGR is used in processing the application. Each script-specific element LGRs presents the complete data and specification to determine the validity of a label as well as to generate the full set of allocatable variants for the label, when applied for under that script.

5.2 Common LGR

The Common (merged) LGR contains the cumulative repertoire, WLE rules and all non-reflexive variant mappings (with type set to “blocked”). The merged LGR thus presents the complete data and specification needed for conflict checking with any existing label in the Root Zone, independent of script.

Note that the merged LGR cannot be used to determine the validity of a label, because the validity of a label depends directly on the specific subset of the overall repertoire that is defined for a given script. (Simply applying the merged LGR would result in returning mixed script labels as valid). The validity of a label may further depend in some circumstances on the script-specific definition of variants. For these reasons, the merged LGR cannot be used for final validity checking of a label.

5.3 Other uses of the Common LGR

As outlined above, the merged LGR serves mainly in the detection of collisions between applied for and delegated (or reserved) labels. In addition, the merged LGR provides:

- documentation of the overall repertoire; in addition to formal data definition in the XML file, and the annotated repertoire table in the HTML, the data from the merge are also used to drive the production of the PDF overview charts;
- documentation of the overall system of WLE rules and actions. The merged rule sets document that rules for different scripts are not in conflict with each other for the same code point;
- an index relating code points to script LGRs; as the *script* from an LGR perspective is not a true partition of the repertoire, particularly for CJK, the common LGR is the way to quickly look up which script LGRs support a code point;
- a starting point for getting from any supported code point in the Root Zone to the description in the various proposal documents and from there to the background documents on which inclusion of these code points is based. To this end, the "ref" attributes identify the relevant proposal for each code point, variant, class, rule and action.

5.4 Steps in Processing a Label

In order to determine the disposition of a label, it is evaluated against the Root Zone LGR in three steps.

1. *Verify that a proposed label is valid by processing it with the Element LGR corresponding to the script that was selected for the label in the application.*

   This check will determine whether all code points in the label are defined in the LGR, and whether each code point meets all the context rules defined for it. In addition, all whole-label rules are evaluated; if a disposition other than "valid" results, the label is invalid.

   At this first step, do not enumerate all variants. However, as part of checking validity it is necessary to evaluate any reflexive variants, and apply dispositions based on their types. For example, if any reflexive variant is of type "out-of-repertoire-var", the label will be invalid.

   For any invalid label, stop the processing.

2. *Process the now validated label against the common LGR to verify it does not collide with any existing delegated labels (and any of their variants, whether blocked or allocatable).*

   Each label and all its variants form a variant label set. For the Root Zone LGR, all variant label relations are symmetric and transitive, which means that all such variant label sets are disjoint (do not overlap). For each label, calculate an Index Label identifying the set (for example the element lowest in code point order). Any two labels resulting in the same index label will collide: either with each other or with one of the variants of the other label. (See also Section 5.5).

For any label that collides with existing labels, stop the processing.

3. *Now that the label is known to be valid, and not in collision, use the appropriate element LGR to generate all allocatable variants.*

   The valid label and any allocatable variants constitute the result of the LGR processing and form the input into any subsequent stages of the application and registration process.

5.5 Index Label Calculation

The most commonly defined variants are those that substitute single code points with neither the code points nor the resulting labels subject to code point context rules or whole-label rules. Where code point context rules or whole-label rules do apply, there may be potential issues in index variant calculation that require careful attention when designing LGRs. In cases where n:m variants are defined (mapping code point sequences of length n to code point sequences of length m), additional complications may arise if n and m share some common code points, or are themselves part of other variant sets (See Section 6.6 “Overlapped Variants”). In these and other cases discussed in Section Design Notes for the Root Zone LGR6, a variant context rule may need to be defined on the variant so it is only defined in situations where the substitution is valid. Otherwise, the resulting sets of variant labels are either not transitive and symmetric, or they may present difficulties in efficient computation of index variants, an essential tool to quickly compute collisions between variant labels.

5.5.1 Background

In order to efficiently detect whether a label is blocked by a variant label, one normally computes a so-called *Index Variant* for both and if they are equal, the two labels are variants of each other. If one has a list of index variants for all registered labels, an application for a new label can be very quickly checked for collisions, as long as the computation of the index variant itself is efficient. To ensure efficient calculation under certain variant set definitions, it is important to be able to calculate the index variant in a single pass (as described below) and still get a correct result. By contrast, any calculation that requires enumerating all variant labels may well be prohibitive, as some longer labels may create very large numbers of blocked variant labels.

5.5.2 Transitivity of Code Point Variant Sets and Variant Label Sets

Transitivity means that all variants in the set are variants of each other. See RFC 8228 for a discussion of this and other concepts related to variants.

For enumerating variants it is strictly required that all allocatable variant labels form a fully transitive variant label set, so that the same set of variants is generated no matter which of the variants is the starting label. For checking collisions, we do not want to have to enumerate all blocked variants – doing so is prohibitive in terms of performance. Therefore, we only require that an LGR be well behaved as far as index label calculation is concerned.

When code point variant sets are defined for code point sequences in LGRs where subsequences of the same sequences are part of the LGR's repertoire (and especially, if they have variants in their own right) then a variant label set may not be transitive, or non-overlapping, even if the code point variant set is defined in a formally transitive manner.

Any LGR with such overlapping sequences requires special attention to ensure that it is well behaved.

5.5.3 Requirements for Index Labels

For the index label method to work, the space of all labels and their variant labels must be divisible into variant label sets so that

1. any label and all its variants belong to the same set
2. no two sets overlap
3. all labels in the set generate the same index label

If these conditions are met, two labels with the same index variant are variants of each other.

For these requirements, it is inessential whether any enumerated variants are also valid labels or not, as long as any invalid labels also belong to only one set.

5.5.4 Generating Index Labels

Index label generation starts with a valid label. (There appears to be no benefit in ensuring that LGRs produce predictable index labels for invalid labels; however, if doing so produces an LGR that can be more easily verified as being correct, there's no reason not to.)

Index label generation proceeds left-to-right in code point sequence. At each point, for any code point or sequence for which a variant is defined, the lowest variant in code point order is substituted (or the original code point or sequence retained if lowest or without variant). If more than one code point/code point sequence start at a give point, an index variant candidate is calculated for each case and the processing continues for that candidate at the end of the given sequence.

This case can arise, for example, if both a sequence and a leading part are separately members of the repertoire. Each division of a label into sequences is called a partition and an index label candidate is produced for each possible partition of the given label. In determining available variants, any variant that has a variant context rule and does not satisfy that rule is ignored. At the end, the lowest candidate becomes the Index Label. If two variants are such that one is a prefix of another, the shorter variant (i.e. the prefix) becomes the Index Label.

Whether or not an index label is a valid label does not matter. In fact, it would be cost-prohibitive to insist that index labels be valid labels: the only way to guarantee that in the general case would be to enumerate all variants and at the end pick the lowest. Many labels have thousands of possible blocked variants (for longer labels the number could be much larger). Therefore, index label generation ignores any code point context rules or whole-label rules.

Note that for the Root Zone, index labels are computed based on the merged or "common" LGR, therefore any "mixed script" labels are notionally in-repertoire.

5.5.5 Impact on Root Zone LGR

For many complex scripts, code point context rules and whole-label rules restrict the set of valid labels. If putative labels are first evaluated against the element LGR to make sure that they would be valid, and then checked against the common (merged) LGR for collisions (as recommended above in Section 5.4), it is not necessary to ensure that invalid labels are well behaved under index variant calculation.

In verifying that the proposed variant definitions were well-behaved for valid labels, it was found that there was a dependency on the choice of index variant: for the Root Zone LGRs, the variant definitions are only well behaved under the assumption that the index label is calculated as described here, using the lowest variant in code point value. Theoretically, an index label could just as well have been calculated using the largest variant, but doing so would require changing or adding some variant definitions.

Therefore, the Root Zone LGR now treats the Index Label Calculation above as a requirement.

6 Design Notes for the Root Zone LGR

6.1 Reducing Complexity

In accordance with the [Procedure] the LGR is designed to mechanically eliminate as much as possible any labels and variant labels that pose an undue risk to the usability and security of the DNS. For many scripts, this requires the use of context or WLE rules to limit the number of valid labels and the use of variants to restrict which labels can be delegated independently.

To reduce complexity of the ruleset, some loss in linguistic fidelity has been accepted where it resulted in simpler rules that do not compromise security. Where possible, constraints have been presented as context rules on code points or via enumeration of sequences in the repertoire. Where context rules are used, those implementing constraints on immediately following or preceding code points have been preferred: no attempt is being made, for example, to implement full segmentation into valid syllables.

Context rules are omitted where they are implicit as result of context rules on all other affected code points in an LGR. Even if a code point has no listed context rules, it may nevertheless have an *implicit* constraint.

As much as possible, the variant mappings and types in the Element LGRs have been drawn up to limit the number of allocatable variants generated. Where applicable, WLE rules reduce the number of valid labels, and in some cases, they reduce the number of allocatable variants as well. Both mechanisms typically rely on dividing the allocatable variants by some suitable linguistic context and then mechanically preventing the mixing of variants from different contexts in the same label.

6.2 Limitations of the LGR

There are limitations to what can be done with mechanical application of rules, and in some cases, it is not possible to reduce the number of allocatable labels in a fashion that is practicable and safe without creating undue restrictions on otherwise valid labels. In this context, it is a useful reminder that having a label that is "allocatable" means neither that it will necessarily be delegated, nor that it necessarily should be delegated. In fact, investigations of actual registrations on the second level reveal that applicants have tended to apply for only a small number of variant labels.

The LGR can be thought of as creating a maximal set of valid labels and allocatable variants, but other steps in the registration process are expected to include suitable mechanisms to further reduce the list of labels available for delegation. It is the view of the Integration Panel that such reduction is necessary, because the larger the number of delegated variants the larger the risk they create to the DNS.

6.2.1 Unicode Version 6.3.0

The design of this version of the Root Zone LGR is based on Unicode 6.3.0, for a discussion see [MSR-4] and [IAB].

6.3 Cross-Script Variants and Security

Many related scripts share character forms so that labels could be constructed wholly within one script, yet indistinguishable from a label in another script. This is an obvious concern for the security of the DNS Root Zone and the IP has been encouraging Generation Panels to identify affected code points and to define them as (blocked) cross-script variants.

The focus is thus on cases where a full label can be created. Cases where the affected scripts only share forms for combining marks could generally be ignored: without a base character, combining marks by themselves cannot form a label.

UCS	Glyph[9]	Name
006F	o	LATIN SMALL LETTER O
03BF	o	GREEK SMALL LETTER OMICRON
043E	o	CYRILLIC SMALLER LETTER O
0585	o	ARMENIAN SMALL LETTER OH
0B20	○	ORIYA LETTER TTHA
0D20	O	MALAYALAM LETTER TTHA
101D	○	MYANMAR LETTER WA

A few very simple shapes, for example the "circle", tend to lack distinguishing features, so that when they occur even in unrelated scripts the IP deems them an unacceptable security risk, unless mitigated.

[9] The choice of fonts may affect the representation of even simple glyphs like this. The shapes shown here are drawn from common user interface fonts.

This risk is exemplified by existing delegation of a .ooo domain in the Root Zone. Establishing blocked variants prevents malicious registrations in other, unrelated scripts, while emphasizing the first-come-first-serve relationship between competing registrations for indistinguishable labels.

6.3.1 Related Scripts and Cross-Script Variants

Normally, the IP attempts to process all related scripts together, but in some cases cross-script variants may exist without a deeper relationship between scripts. In such cases, there may be proposed variants between scripts that are not processed concurrently, so that, for example, the two GPs for the affected scripts are not in session at the same time, or do not produce concurrent drafts. A similar case may arise from deferred scripts, for which a GP may no longer be constituted at the time they are finally integrated.

Proposed variants to scripts not yet in the current LGR version (and not concurrently processed) cause an issue in integration because the integrated LGR must have transitive closure, yet cannot contain code points that are outside the collective repertoire. If an LGR contains such variants to a “future” script, they would have to be deferred temporarily from the integrated LGR until such time that the future script is finally added.

To facilitate this process, GPs are encouraged to list such variants, if possible, separately in their LGR document as tentative variants, but not to list them in the XML specification of the proposal. The IP plans to take any such listings of tentative variants to scripts not in the current LGR version and note them in this "overview" document for the next published LGR. The IP will then work with the GP for the future script based on that list to make them part of the future proposal. If necessary, the IP will raise the issue during public comments on the future script and depending on the outcome of that process make them a requirement for acceptance of the proposal. Where a future script is in the early stages and may already have a GP seated, the GPs are also encouraged to engage in dialog; any seated GP for a future script is encouraged to comment on any tentative cross-script variants in an LGR under public comment.

In case of unrelated scripts (e.g. out of region, without or with less direct historical derivation) GPs have been reluctant to identify certain critical cross script variants. The IP reiterates that security-relevant true homoglyphs are in scope for cross-script variants, independent of whether the scripts are related.

In order to assure a secure Root Zone, the IP has identified some these critical cases, such as the “circle” (see Section 6.3). The IP plans to work with affected GPs to ensure that they are included; this may include raising notice in public comment for any affected LGR, and if necessary rejecting proposals that omit variants that are deemed critical for a secure DNS,

In case where finalized LGR proposals differ in cross-script variants for any reason, the IP will try to get GPs to resolve any differences, but where that is not possible, the IP will resolve these as prescribed in the Procedure. The Procedure prescribes a mechanical integration process that creates the union and transitive closure for these variants as part of integration - provided that this does not lead to in-script variants in any of the affected scripts (other than in the special case of overlapping repertoires).

6.3.2 Transitive Closure

Transitive Closure is defined on the code point level and in order to enforce it across the entire Root Zone LGR during integration, some mappings may need to be defined for certain scripts on the code point level even if no label could be built. (This can happen if one script has both independent and dependent code points as variants with two scripts, and with the dependent variant the same code point. If those two scripts do not have any independent code point as variant between them, transitivity still requires that the "orphaned" dependent code point is mapped between the two scripts - counter to the general policy requiring a base character).

6.4 Code Point Sequences

An LGR may contain both single code points as well as sequences in its repertoire. Any code point that exists only as a member of a sequence, but is not listed otherwise in the repertoire may be part of any label as part of that sequence, but not otherwise. Sequences are thus a mechanism for enumerating limited numbers of allowable combinations, such as for base characters and diacritics. Such enumerations are considered the most “light-weight” constraints on labels, and therefore preferable to other types of constraints on labels.

6.4.1 Sequences and Context Rules

Any context rules defined for code points or subsequences are not evaluated if these occur inside a larger sequence. For example, a sequence that starts with a code point that may only follow consonants does not automatically inherit that restriction. A sequence may sometimes be defined intentionally to *override* a context restriction otherwise defined for a certain code point in the context of that sequence. Sequences for which such an override is not intended, must be given a context that restricts them to the same positions in the label as equivalent combinations of code points taken as singletons.

Conversely, the presence of a restrictive code point context on a sequence may be ineffective, as long as any contexts defined on the individual code points allow them to be used in the same combination as they occur in the sequence.

The preceding discussion also applies to any subsequences that are not singletons, but listed as members of the repertoire.

6.4.2 Sequences Defined For Use as Variants

A sequence may be defined solely as a target for a variant mapping. In that case, the Root Zone LGR generally restricts the contexts the sequence may occur in to contexts for which the variant mapping should be available. If that is not possible, context constraints may be defined for the variant mapping itself. By reasons of symmetry, both forward and reverse mapping must have the same variant context. See [RFC8228] for details.

Where both a sequence and some subsequence independently have variant mappings, they are said to *overlap* and special care was taken to ensure that the overall system of variant labels is well behaved.

A particular type of variant that requires context rules on both sequences and variant mappings is discussed in the following section.

6.5 Effective Null Variants

A *Null Variant* is defined in [RFC7940] as a variant mapping from a code point to an empty position. Such variants are not deemed well behaved for purposes of the Root Zone as they would define a variant for any position between any two code points.

Variant definitions where a sequence is mapped to a shorter sequence which is at the same time contained in the original sequence (for example where the shorter sequence is a prefix of the longer one) are very similar to null variants.

For example,

AB → A

and the symmetric (reverse) mapping

A → AB

are logically equivalent to a Null Variant with a context rule

B → ∅ : when(preceded-by-A)

∅ → B: when(preceded-by-A)

Such *effective null variants* are also not well-behaved: each label in a variant label set containing such an effective null variant would have additional variant labels that are longer.

A has variants A, AB
AB has variants A, AB, ABB

and so on, with the original label underscored. The set of variant labels are no longer disjoint, but overlap instead. In mathematical terms, there is no *transitive closure*.

However, the addition of a formal context rule on such variants can make them well behaved. The context rule needs to ensure that any variant label already containing the longer sequence cannot be "expanded" by applying the variant mapping to the shorter (contained) sequence.

For example:

A → AB : when-not(followed by B)

AB → A : when-not(followed by B)

With this additional constraint, the label AB does not have a variant ABB, therefore:

AB has variants A, AB.

The real-world case for this exists, for example, in Devanagari, where there is a desire to treat code points with and without NUKTA (a dot below) as variants, because not all parts of the community would

recognize a NUKTA as a distinguishing feature. In scripts where diacritical marks are precomposed, comparable variant mappings often become simple 1:1 mappings between single code points with and without the diacritic. This would avoid the complications described here.

Effective Null Variants exist for any common subsequence, even if the sequence is not contiguous.

For example:

CHC → CC

is equivalent to

H →Ø : when(preceded-and-followed-by-C).

As in the earlier example, the addition of a context on the variant mapping would make it well behaved:

CHC → CC : when-not(followed-by-C).

Note that for label CCC, the above constraint would limit the variants to CCHC, and not recognize CHCC as a variant. The real-world case for this exists in Malayalam and additional sequences needed to be defined to handle longer sequences. Wherever possible, the Root Zone LGR prefers to disallow some rare labels instead of admitting the complexity of effective null variants, but this is not always possible for complex scripts.

6.6 Overlapped Variants

Null variants and effective null variants are both examples of *overlapped* variants. For overlapped variants, part of one side of a variant mapping has its own, unrelated, variant mapping.

For example:

AB → C

A → D

When calculating the variants for AB all possible partitions are considered. In this case, assuming B is also an element of the repertoire on its own, the partitions would be {AB} and {A}{B}. Including the original code points the variant sets would be:

{AB} → AB, C

{A}{B} → AB, DB

While both C and DB have a reverse mapping to AB, there's no mapping between them, and the variant set is no longer transitive. In some cases, adding the missing mappings

AB → DB

C → DB

would make the set transitive. Actual examples of this can be found in the Devanagari and Sinhala LGRs. In those cases, the new mappings are not only formally required to make the set well behaved, but also reflected real variant relations.

7 Summary of Changes

7.1 Changes by revision

1. LGR-1 added 128 code points for 1 script, plus 17 WLE rules and 21 actions.
2. LGR-2 added 535 code points for 5 scripts, plus 27 WLE rules and 1 action.
3. LGR-3 added 655 code points for 10 scripts, plus 44 WLE rules and 2 action(s).

7.2 Code points by script

The following table shows how many code points, by script, are available for root zone LGR development by being included in [MSR-4] and how many are selected for each version of the LGR. The count includes code points that are only available as part of a defined sequence.

Script tag	Script Name	MSR-4	LGR-1	LGR-2	LGR-3
Arab	Arabic	239	128	128	128
Armn	Armenian	38			
Beng	Bengali	64			
Cyrl	Cyrillic	93			
Deva	Devanagari	91			84
Ethi	Ethiopic	364		311	311
Geor	Georgian	37		33	33
Grek	Greek	36			
Gujr	Gujarati	66			65
Guru	Gurmukhi	61			56
Hang	Hangul	11 172			
Hani	Han Ideographs	19 855			
Hebr	Hebrew	46			27
Hira	Hiragana	89			
Kana	Katakana	92			
Khmr	Khmer	78		71	71
Knda	Kannada	68			62
Laoo	Lao	53		51	51
Latn	Latin	311			
Mlym	Malayalam	73			70
Mymr	Myanmar	102			
Orya	Oriya	66			62
Sinh	Sinhala	79			72
Taml	Tamil	49			48
Telu	Telugu	67			63
Thaa	Thaana	50			
Thai	Thai	71		69	69
Tibt	Tibetan	80			
Zinh	INHERITED	21			
Total		**33 511**	**128**	**663**	**1 318**

8 Contributors

LGR-3 and its precursor versions were developed by the Integration Panel, based on proposals submitted by the respective Generation Panels, with input from community members, as well as support by ICANN staff members. The following lists of contributors are cumulative.

8.1 Integration Panel Members

Marc Blanchet
Asmus Freytag
Nicholas Ostler
Michel Suignard
Wil Tan

8.2 Advisors

Lu Qin

8.3 Community Members

The Integration Panel gratefully acknowledges the information provided by the following members of the community:

Members of TF-AIDN (Arabic) [TF-AIDN]
Members of the Armenian Generation Panel [Armenian GP]
Members of the Ethiopic Generation Panel [Ethiopic GP]
Members of the Georgian Generation Panel [Georgian GP]
Members of the Hebrew Generation Panel [Hebrew GP]
Members of the Khmer Generation Panel [Khmer GP]
Members of the Lao Generation Panel [Lao GP]
Members of the Neo-Brahmi Generation Panel [NeoBGP]
Members of the Sinhala Generation Panel [Sinhala GP]
Members of the Thai Generation Panel [Thai]
Olivier Crepin-Leblond
Chris Dillon
Liang Hai
Yuriy Kargapolov
Narine Khachatryan
Meikal Mumin
Dusan Stojicevic
Richard Wordingham

8.4 ICANN Staff

Sarmad Hussain
Pitinan Kooarmornpatana
Alireza Saleh
Anand Mishra
Jia-Juh Kimoto
Kim Davies

9 References

[Armenian GP] Armenian Script Generation Panel, see Section 8 of [Proposal-Armenian]

[Cyrillic GP] Cyrillic Generation Panel, see section 8 of [Proposal-Cyrillic]

[Ethiopic GP] Ethiopic Script Generation Panel, see Section 8 of [Proposal-Ethiopic]

[Georgian GP] Georgian Script Generation Panel, see Section 8 of [Proposal-Georgian]

[Hebrew GP] Hebrew Script Generation Panel, see Section 8 of [Proposal-Hebrew]

[Khmer GP] Khmer Generation Panel, see Section 8 of [Proposal-Khmer]

[Lao GP] Lao Generation Panel, see Section 8 of [Proposal-Lao]

[NeoBGP] Neo-Brahmi Generation Panel, see Sections 4 and 8 of [Proposal-Devanagari], [Proposal-Gujarati], [Proposal-Gurumukhi], [Proposal-Kannada], [Proposal-Malayalam], [Proposal-Oriya], [Proposal-Tamil], and [Proposal-Telugu]

[Sinhala GP] Sinhala Generation Panel, see Section 8 of [Proposal-Sinhala]

[Thai GP] Thai Generation Panel, see Section 8 of [Proposal-Thai]

[Guidelines] Internet Corporation for Assigned Names and Numbers, “Guidelines for Developing Script-Specific Label Generation Rules for Integration into the Root Zone LGR”. (Los Angeles, California: ICANN, December 2014) https://community.icann.org/download/attachments/43989034/Guidelines-for-LGR-2014-12-02.pdf.

[IAB] Internet Architecture Board (IAB), "IAB Statement on Identifiers and Unicode 7.0.0" https://www.iab.org/documents/correspondence-reports-documents/2015-2/iab-statement-on-identifiers-and-unicode-7-0-0/

[IABCP] Sullivan, A., *et al.,* “Principles for Unicode Code Point Inclusion in Labels in the DNS”. Internet Architecture Board (IAB) = RFC 6912 http://tools.ietf.org/html/rfc6912.

[IAB-Comment] Sullivan, A., “Comments from the IAB on LGRs for second level”, 17 July 2016, https://forum.icann.org/lists/comments-lgr-second-level-07jun16/msg00001.html

[IDNAREG] IANA Registry: "IDNA Parameters". For Unicode 6.3 available at: http://www.iana.org/assignments/idna-tables-6.3.0/idna-tables-6.3.0.xml Visited 2019-06-05.

[ISO15924] *Codes for the representation of names of scripts*, ISO 15924:2004. Available from http://www.unicode.org/iso15924/. Visited 2012-06-05.

[LGR-1] Integration Panel, “Integration Panel: Root Zone Label Generation Rules — LGR-1”, 24 February 2016, https://www.icann.org/sites/default/files/lgr/lgr-1-overview-24feb16-en.pdf

[LGR-2] Integration Panel, “Integration Panel: Root Zone Label Generation Rules — LGR-2”, 26 July 2017, https://www.icann.org/sites/default/files/lgr/lgr-2-overview-26jul17-en.pdf

[MSR-4] Integration Panel, "Maximal Starting Repertoire — MSR-4 Overview and Rationale", 7 February 2019 https://www.icann.org/en/system/files/files/msr-4-overview-25jan19-en.pdf [PDF, 0.8 MB]

[Packaging] Integration Panel: “Packaging the MSR and LGR”, 24 April 2015, https://community.icann.org/download/attachments/43989034/Packaging-MSR-LGR.pdf

[Procedure] Internet Corporation for Assigned Names and Numbers, "Procedure to Develop and Maintain the Label Generation Rules for the Root Zone in Respect of IDNA Labels." (Los Angeles, California: ICANN, March, 2013) http://www.icann.org/en/resources/idn/variant-tlds/draft-lgr-procedure-20mar13-en.pdf

[Proposal-Arabic] TF-AIDN, "Proposal for Arabic Script Root Zone LGR", Version 3.4, 18 November 2015, Supporting Document: https://www.icann.org/en/system/files/files/arabic-lgr-proposal-18nov15-en.pdf [PDF, 3.47 MB]

[Proposal-Armenian] Armenian Generation Panel, "Proposal for an Armenian Script Root Zone LGR", 05 November 2015, Supporting Document: https://www.icann.org/en/system/files/files/armenian-lgr-proposal-05nov15-en.pdf [PDF 1.04MB]

[Proposal-Cyrillic] Cyrillic Generation Panel, "Proposal for Cyrillic Script Root Zone Label Generation Rules", 03 April 2018, Supporting Document: https://www.icann.org/en/system/files/files/proposal-cyrillic-lgr-03apr18-en.pdf [PDF 1.2MB]

[Proposal-Devanagari] Neo-Brahmi Generation Panel, "Proposal for the Devanagari Script Root Zone LGR", 22 April 2019, Supporting Document: https://www.icoan.org/en/system/files/files/proposal-devanagari-lgr-22apr19-en.pdf [PDF 1.6 MB]

[Proposal-Ethiopic] Ethiopic Script Generation Panel, “Proposal for Ethiopic Script Root Zone LGR”, 17 May 2017, Supporting Document: https://www.icann.org/en/system/files/files/proposal-ethiopic-lgr-17may17-en.pdf [PDF, 2.01 MB]

[Proposal-Georgian] Georgian Script Generation Panel, “Proposal for the Georgian Script Root Zone LGR”, 24 November 2016, Supporting Document: https://www.icann.org/en/system/files/files/proposal-georgian-lgr-24nov16-en.pdf[PDF, 474 KB]

[Proposal-Gujarati] Neo-Brahmi Generation Panel, "Proposal for the Gujarati Script Root Zone LGR", 6 March 2019, Supporting Document: https://www.icann.org/en/system/files/files/proposal-gujarati-lgr-06mar19-en.pdf [PDF 2.29 MB]

[Proposal-Gurmukhi] Neo-Brahmi Generation Panel, "Proposal for the Gurmukhi Script Root Zone LGR", 22 April 2019, Supporting Document: https://www.icann.org/en/system/files/files/proposal-gurmukhi-lgr-22apr19-en.pdf [PDF 364 KB]

[Proposal-Hebrew] Hebrew Generation Panel, "Proposal for a Hebrew Script Root Zone Label Generation Ruleset (LGR)", 24 April 2019, Supporting Document: https://www.icann.org/en/system/files/files/proposal-hebrew-lgr-24apr19-en.pdf [PDF 403 KB]

[Proposal-Kannada] Neo-Brahmi Generation Panel, "Proposal for the Kannada Script Root Zone LGR", 6 March 2019, Supporting Document: https://www.icann.org/en/system/files/files/proposal-kannada-lgr-06mar19-en.pdf [PDF 2.24MB]

[Proposal-Khmer] Khmer Generation Panel, “Proposal for Khmer Script Root Zone Label Generation Rules (LGR)”, 15 August 2016, https://www.icann.org/en/system/files/files/proposal-khmer-lgr-15aug16-en.pdf [PDF, 3.26 MB]

[Proposal-Lao] Lao Generation Panel, "Proposal for a Lao Script Root Zone LGR", January 31, 2017, Supporting Document: https://www.icann.org/en/system/files/files/proposal-lao-lgr-31jan17-en.pdf [PDF 2.2MB]

[Proposal-Malayalam] Neo-Brahmi Generation Panel, "Proposal for the Malayalam Script Root Zone LGR", 22 April 2019, Supporting Document: https://www.icann.org/en/system/files/files/proposal-malayalam-lgr-22apr19-en.pdf [PDF 782 KB]

[Proposal-Oriya] Neo-Brahmi Generation Panel, "Proposal for the Oriya Script Root Zone LGR", 6 March 2019, Supporting Document: https://www.icann.org/en/system/files/files/proposal-oriya-lgr-06mar19-en.pdf [PDF 1.63 MB]

[Proposal-Sinhala] Neo-Brahmi Generation Panel, "Proposal for the Sinhala Script Root Zone LGR", 22 April 2019, Supporting Document: https://www.icann.org/en/system/files/files/proposal-sinhala-lgr-22apr19-en.pdf [PDF 855 KB]

[Proposal-Tamil] Neo-Brahmi Generation Panel, "Proposal for the Tamil Script Root Zone LGR", 6 March 2019, Supporting Document: https://www.icann.org/en/system/files/files/proposal-tamil-lgr-06mar19-en.pdf [PDF 2.83 MB]

[Proposal-Telugu] Neo-Brahmi Generation Panel, "Proposal for the Telugu Script Root Zone LGR", 6 March 2019, Supporting Document: https://www.icann.org/en/system/files/files/proposal-telugu-lgr-06mar19-en.pdf [PDF 1.0 MB]

[Proposal-Thai] Thai Generation Panel, "Proposal for the Thai Script Root Zone LGR", 25 May 2017, Supporting Document:https://www.icann.org/en/system/files/files/proposal-thai-lgr-25may17-en.pdf

[RFC5890] Klensin, J., "Internationalized Domain Names for Applications (IDNA): Definitions and Document Framework", RFC 5890, August 2010.

[RFC5891] Klensin, J., "Internationalized Domain Names in Applications (IDNA): Protocol", RFC 5891, August 2010.

[RFC5892] Faltstrom, P., Ed., "The Unicode Code Points and Internationalized Domain Names for Applications (IDNA)", RFC 5892, August 2010.

[RFC5893] Alvestrand, H., Ed., and C. Karp, "Right-to-Left Scripts for Internationalized Domain Names for Applications (IDNA)", RFC 5893, August 2010.

[RFC6912] Sullivan, A., *et al.,* "Principles for Unicode Code Point Inclusion in Labels in the DNS", RFC 6912, April 2013. = IABCP

[RFC7940] Davies, K. and A. Freytag, "Representing Label Generation Rulesets using XML", RFC 7940, August 2016, https://tools.ietf.org/html/rfc7940

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

[TF-AIDN] Blog, "Task Force for Arabic Script IDNs"; see Appendix H of [Proposal-Arabic] for members that contributed to the development of the Arabic Script LGR Proposal.

[UAX24] UAX #24: *Unicode Script Property.* An integral part of The Unicode Standard. Most recent version available from http://www.unicode.org/reports/tr24/. Version 6.3 available as http://www.unicode.org/reports/tr24/tr24-21.html.

[Unicode63] The Unicode Consortium. The Unicode Standard, Version 6.3.0, defined by: "The Unicode Standard, Version 6.3.0", (Mountain View, CA: The Unicode Consortium, 2013. ISBN 978-1-936213-08-5). http://www.unicode.org/versions/Unicode6.3.0/.

[UCD] UAX #44: *Unicode Character Database.* An integral part of The Unicode Standard. Most recent version available from http://www.unicode.org/reports/tr44/. Version 6.3 available as http://www.unicode.org/reports/tr44/tr44-12.html.