Label Generation Rulesets (LGRs) specify metadata, code point repertoire, variant rules and Whole Label Evaluation (WLE) rules to generate labels.

RFC 7940 describes how LGR can be specified using XML, a machine readable format.

LGR can be used to generate domain name labels for use in the internet’s root zone and other levels.

LGR Toolset can be used to:
- create an LGR
- view LGR as an HTML webpage or XML Format
- merge multiple LGRs into a single LGR
- validate single label or multiple labels using an LGR
- determine cross-script variants of labels using a merged LGR
- manage LGRs by comparing or combining them
- review possible impact of a new or a revised LGR on existing labels
- harmonize multiple LGRs
- compute variants
Availability of LGR Toolset

- LGR Toolset is available with the following disclaimer:

  THIS SOFTWARE IS PROVIDED BY ICANN AND CONTRIBUTORS ``AS IS'' AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL ICANN OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

- Online deployment
  - Visit https://lgrtool.icann.org/

- Open source package(s) release with BSD license
  - Released at github: lgr-core, lgr-django, munidata, picu

- For queries or feedback
  - Email to IDNProgram@icann.org

- For further details, visit the LGR Toolset webpage or www.icann.org/idn
Agenda

1. Landing Page (pp. 11-12)
2. Advanced LGR Tools (pp. 13-14)
3. Import or Load LGR (pp. 15-18)
4. Importing Multiple LGRs (pp. 19-22)
5. Create LGR (pp. 23-49)
6. Validate LGR (pp. 50-53)
Agenda (Cont’d)

7. Viewing LGR as XML and HTML (pp. 54-65)

8. Validating a Label (pp. 66-71)

9. Review Impact on Existing Labels by Revising an Existing LGR (pp. 72-77)

10. Review Impact on Existing Labels by Introducing a New LGR (pp. 78-83)

11. Validating Multiple Labels Using a Single or Merged LGR(s) (pp. 84-90)

12. Viewing Cross-Script Variants of Labels (pp. 91-96)
Compare LGRs (pp. 97-101)

Harmonize Multiple LGRs (pp. 102-105)

Compute Variants (pp. 106-110)

Label forms (pp. 111-118)
Agenda Details

- Landing Page
- Advanced LGR Tools
- Import or Load LGR
  - Import LGR
  - Select LGR with validating repertoire
- Import Multiple LGRs
  - Import LGR
  - Select multiple LGRs with validating repertoire
  - Specify name of the merged LGR
Agenda Details (Cont’d)

- **Create LGR**
  - Create LGR
  - Enter LGR Details
  - Add Code Points
  - Expand Ranges
  - Add References
  - Define Metadata
  - Define Classes, Rules & Actions
  - Define Code Point Properties
  - Download LGR

- **Validate LGR**

- **View LGR as XML and HTML**

- **Validate a Label**
  - Validate a label using a single LGR
  - Validate a label using a merged LGR
Agenda Details (Cont’d)

◉ Review Impact on Existing Labels by Revising an Existing LGR
  o Select “Diff labels of two LGRs”
  o Enter Details
  o Email Notification
  o Download Results

◉ Review Impact on Existing Labels by Introducing a New LGR
  o Select “Get collisions in a list of labels”
  o Enter Details
  o Email Notification
  o Download Results

◉ Validate Multiple Labels Using a Single or Merged LGR(s)
  o Select “Generate disp. annotations”
  o Enter Details
  o Email Notification
  o Download Results
Agenda Details (Cont’d)

✎ View Cross-Script Variants of Labels
  o Select “Cross-script variants”
  o Enter Details
  o Email Notification
  o Download Results

✎ Compare LGRs
  o Compare LGRs
  o Select Union, Intersection or Difference
  o With merged LGRs, “Diff” chosen by default

✎ Harmonize Multiple LGRs
  o Select “Harmonization”
  o Enter Details
  o Email Notification
  o Download Results

✎ Compute Variants
  o Select “Compute Variants”
  o Select relevant LGR
  o Upload list of labels and enter email
  o Download results on email notification

✎ Label Forms
Landing Page
Landing Page of the LGR tool

- A user guide with more information is published on the [LGR Tool webpage](#).
- Go to link - Use the **Label Generation Rules Tool**

**Welcome to the LGR (Label Generation Ruleset) Tools**

Select your mode:

- **Basic Mode**
  - Validate label(s) against an LGR
- **Advanced Mode**
- **IDN Table Reviewing**
Advanced LGR Tools
Advanced LGR Tools

Advanced LGR (Label Generation Ruleset) Tools

This application provides a convenient interface for browsing and editing LGRs conforming to the Representing Label Generation Rulesets using XML specification.

Previously loaded LGR file(s)
No LGR has been previously loaded.

Create a new LGR file or import an existing one

- Import an existing XML file

⚠️ Note that importing large LGR files may take significant time to load on your browser.

- Start with a New blank XML file

Start from a built-in LGR
The following LGRs are pre-installed in the system. You may use them as a starting point for your own LGR. To do so, just click on it to make a copy that you can then edit.

- Open Sample-French
- Open RZ-LGR 1
- Open RZ-LGR 2
- Open RZ-LGR 3
- Open RZ-LGR 4
- Open RZ-LGR 5

Remember to save your work regularly by downloading a copy of the XML file.

Please send any feedback to support@viagenie.ca.

Your saved results
The following files contains your tools computation results.

⚠️ Note that these files could be cleaned up regularly.

- Download 20221022_171249_annotation_RZ-LGR_5.txt.gz
Import or Load LGR
To start by using an existing LGR file in XML format, click on the “Import” button.
Select LGR with Validating Repertoire

1. To import or load an existing LGR in XML format, click on “Choose Files”

2. Choose the “Validating repertoire” from the given options

3. Click on the “Import” button
Imported LGR

The screen looks like this after successful import of existing LGR file in XML format.
Import Multiple LGRs
Import or Load LGRs

To start by using existing LGR files in XML format, click on the “Import” button.

Previously loaded LGR file(s)
No LGR has been previously loaded.

Create a new LGR file or import an existing one

- Import an existing XML file
- Start with a New blank XML file

Note that importing large LGR files may take significant time to load on your browser.
Select LGRs with Validating Repertoire

1. To import or load multiple LGRs in XML format, click on “Choose Files” and select multiple files.
2. Choose the “Validating repertoire” from the given options.
3. Enter a name for the set of LGRs.
4. Click on the “Import” button.
Imported LGRs As a Merged LGR

The screen looks like this after successful import of multiple LGR files in XML format.
Create LGR
Create an LGR

To start by creating an LGR file in XML format, click on the “New” button.

Previously loaded LGR file(s)
No LGR has been previously loaded.

Create a new LGR file or import an existing one

- Import an existing XML file

⚠️ Note that importing large LGR files may take significant time to load on your browser.

- Start with a New blank XML file
Enter LGR Details

1. Write name for the LGR being created

2. Select Validating repertoire from the given options for the “New” LGR. MSR for RZ-LGR and IDNA version for second level LGRs

3. Click on the “Create” button
Click on “Add code points” to add code points to the newly created LGR.
To add code points, there are four different ways:

1. Add code points one by one
2. Add code points by giving a range
3. Add code points by specifying a script
4. Add code points from a file
Add Code Points

1. Write the code point to be added. The code point value or the actual character can be entered.

2. Click on the “Add Code Point” button.
Add Code Points

1. Add the first code point of the range

2. Add the last code point of the range

Either code point value or the character can be added

3. Click on “Next” button
Add Code Points

1. Select a script from the given options
2. Select Validating Repertoire from the given options
3. “Manual import” is optional – allows checking each code point in the script before adding to the LGR
4. Click on “Next” button
Add Code Points

1. Select the file containing code points
2. Select the file type from the given options
3. “Manual import” is optional – allows checking each code point in the file before adding to the LGR
4. Click on the “Next” button
Expand Ranges

To expand all the code points in all the ranges of the LGR, click on “Expand range(s)”.

To expand all the code points in this range, click on “Expand range”.

[Diagram showing a table of code points and actions like 'Expand range' and 'See code point']
Add References

1. Add Reference id
2. Add comments
3. Add detailed reference
4. Click on the “Add” button

Click on “References” tab to add references to the LGR
Add References

“Existing references” tab shows the added references
Define Metadata

Click on “Metadata” tab to add meta information about the LGR.
Define Meta Data

Second half of the “Metadata” tab

Add any text for describing the LGR

Select “Description type” from the given options

Select “Validating repertoire” from the given options

After filling every detail, click on the “Save” button
Click on “Tags” tab to view all tag names and associated code points.
Define Classes, Rules & Actions

Click on “Rules” tab to add certain rules to the LGR

Click on “New class” button to add classes to the LGR

Click on “New rule” button to add rules to the LGR

Click on “New action” button to add actions to the LGR
Define Classes

1. Add classes in the relevant box

2. Click on the “Save” button
Define Rules

1. Add the rule in the relevant box
2. Click on the “Save” button
Define Actions

1. Add action in the relevant box
2. Click on the “Save” button
Apply Batch Action

1. Select multiple code points to apply the batch action
2. Select the type of batch action
Apply Batch Action – Add WLE

1. Add when-rule / not-when-rule from the list of WLE rules
2. Click on “Next” button
Apply Batch Action – Add Tags

1. Type new tags, separate by a space, or select the existing ones
2. Click on “Next” button
Define Code Point Properties

Click on the “See code point” button to add code point details/properties.
Define Code Point Properties - 1

1. Add variant for the code point
2. Click on the “Add variant button”
3. Add details for the added variant – type, comments, when-rule and not-when rule
4. Add tags for the code point
5. Add when-rule/not-when rule from the list of rules it provides
6. Add any description for the code point
Define Code Point Properties - 2

7. Click on “Save variants, tags, context rules and comment” button

Note: If the added code point is not in the repertoire, the system will automatically add the out-of-repertoire mapping.

8. Click on the “Edit” button to add references to the code point

To delete code point & its details, click on the “Delete code point” button
Define Code Point Properties - 3

The screen looks like this after successfully defining code point properties.

If the variant definitions are not symmetric, this button will show up. Click “Populate variants” to automatically populate variant mappings.
Click on “Output” and select “Download” from the given options to download the created LGR.
Validate LGR
Click on “Validate LGR” button to validate LGR and get a summary of the entire LGR. It is important to note that this function should be used for checking an LGR before use, e.g. when it is created or imported.
Validate LGR

Summary of checks performed, including symmetry and transitivity.
Validate LGR

Second half of the "Validate LGR" output

Summarized LGR
View LGR as XML and HTML
View LGR As XML

Click on “Output” and select “View XML” from the given options to get an XML view of the LGR.
This XML file does not appear to have any style information associated with it. The document tree is shown below.

```xml
<lg xmlns="urn:ietf:params:xml:ns:lg-1.0">
  <meta>
    <version comment="Thai Script Root Zone LGR Version 6.9">2</version>
    <date>2017-05-25</date>
    <unicode-version>6.3.0</unicode-version>
    <language>und-Thai</language>
    <scope type="domain"></scope>
  </meta>
  <![CDATA[
    <h1>Label Generation Rules for the Thai Script</h1>
    <h2>Overview</h2>
    <p>This file contains Label Generation Rules (LGR) for the Thai script as would be appropriate for the Root zone. For more details on this LGR see "Proposal for a Thai Script Root Zone LGR [Proposal]".
  ]]>[
  <h2>Repertoire</h2>

  2> In addition to the 68 code points according to Section 5 “Repertoire” in [Proposal], three sequences have been defined. The sequence U+0E4D U+0E32 was defined to replace the disallowed U+0E33 (THAI CHARACTER SARA AM) and to facilitate implementation of WLE rule <b>follows-consonant-tone</b> as a context rule. The other two sequences were defined to restrict U+0E45 (THAI CHARACTER LAKKHANGYAO) from appearing in any context other than these sequences. Accordingly, while U+0E45 is not listed by itself it brings the total of distinct code points to 69.

  2> Variants

  According to Section 6 "Variants", in "[Proposal]", this LGR defines no variants.

  2> Character Classes

  The Thai Script is an abugida in which consonant-vowel sequences are written as a unit: each unit is based on a consonant letter, and vowel, tone mark or diacritic notation are secondary. It is written with the combining marks stacked above or below the base consonant, like diacritics in European languages. However, although the concepts are quite similar, the implementations are significantly different.

  There are 44 characters that are classified as consonants, code points from this subset have been given the tag "cons".

  The 18 vowel symbols pronounced after a consonant are non-sequential: they can be located before (lv), after (fv), above (av) or below (bv) the consonant, or in a combination of these positions, code points from this subset have been given the tag "fv1", "fv2", "fv3", "av", "bv", "lv". There are three code point sequences defined that include vowels. (Code point sequences do not carry tag values; instead, for code point sequences the subset values are indentified in comments).

  There are 5 phonemic tones: mid, low, falling, high, and rising. These 5 tones are represented by 4 tone marks plus the absence of a mark. Code points from this subset have been given the tag "tone". They differ in their frequency and purpose of usage. See also the discussion in section 5.4 in [Proposal].

  There are 3 diacritic symbols that have been included here and given the tag "ad". They differ in their frequency and purpose of usage. See also the discussion in section 5.4 in [Proposal].

  MAITAIKHU and U+0E4C (THANTHAKHAT) are commonly used in everyday communicating words.</li>]
</lg>
```
Click on “Output” and select “HTML Output” from the given options to view the HTML output of the LGR.
Table of Contents

1 Description
2 Repertoire
3 Variant Sets
4 Classes, Rules and Actions
   4.1 Character Classes
   4.2 Whole label evaluation and context rules
   4.3 Actions
5 Table of References

Description
Label Generation Rules for the Thai Script

Overview

This file contains Label Generation Rules (LGR) for the Thai script as would be appropriate for the Root zone. For more details on this LGR see "Proposal for a Thai Script Root Zone LGR [Proposal]."

Repertoire

In addition to the 68 code points according to Section 5 “Repertoire” in [Proposal], three sequences have been defined. The sequence U+0E4D U+0E32 was defined to replace the disallowed U+0E33 (THAI CHARACTER SARA AM) and to facilitate implementation of WLE rule follows-consonant-tone as a context rule. The other two sequences were defined to restrict U+0E45 (THAI CHARACTER LAKKHANYAO) from appearing in any context other than these sequences. Accordingly, while U+0E45 is not listed by itself it brings the total of distinct code points to 69.

Variants

According to Section 6 "Variants", in "$[Proposal]$, this LGR defines no variants.

Character Classes

The Thai Script is an abugida in which consonant–vowel sequences are written as a unit: each unit is based on a consonant letter, and vowel, tone mark or diacritic notation are secondary. It is written with the combining marks stacked above or below the base consonant, like diacritics in European languages. However, although the concepts are quite similar, the implementations are significantly different.

There are 44 characters that are classified as consonants, code points from this subset have been given the tag "cons".

The 18 vowel symbols pronounced after a consonant are non-sequential: they can be located before (lv), after (fv), above (av) or below (bv) the consonant, or in a combination of these positions, code points from this subset have been given the tag "fv1", "fv2", "fv3", "av", "bv", "lv". There are three code point sequences defined that include vowels. (Code point sequences do not carry tag values; instead, for code point sequences the subset values are identified in comments).

There are 5 phonemic tones: mid, low, falling, high, and rising. These 5 tones are represented by 4 tone marks plus the absence of a mark. Code
### HTML Output - Repertoire

**Repertoire**

**Summary**

<table>
<thead>
<tr>
<th>Number of elements in repertoire</th>
<th>434</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of ranges in repertoire</td>
<td>0</td>
</tr>
<tr>
<td>Number of code point sequences</td>
<td>4</td>
</tr>
</tbody>
</table>

**Repertoire by Code Point**

The following table lists the repertoire by code point (or code point sequence). The data in the Script and Name column are extracted from the Unicode character database. Where the comment in the original LGR is equal to the character name, it has been suppressed.

For any code point or sequence for which a variant is defined, the link to the associated variant set, or if mapped to itself, the variant type of that mapping is provided in the Variants column.

<table>
<thead>
<tr>
<th>#</th>
<th>Code Point</th>
<th>Glyph</th>
<th>Script</th>
<th>Name</th>
<th>Tags</th>
<th>Required Context</th>
<th>Variants</th>
<th>Comment</th>
<th>References</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>U+0E01</td>
<td>น</td>
<td>Thai</td>
<td>THAI CHARACTER KO KAI</td>
<td>Thai,und-Thai-cons</td>
<td></td>
<td>[5], [100], [101]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>U+0E02</td>
<td>ข</td>
<td>Thai</td>
<td>THAI CHARACTER KHO KHAI</td>
<td>Thai,und-Thai-cons</td>
<td></td>
<td>[5], [100], [101]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>U+0E03</td>
<td>ฃ</td>
<td>Thai</td>
<td>THAI CHARACTER KHO KHUAT</td>
<td>Thai,und-Thai-cons</td>
<td></td>
<td>[5], [100], [101]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>U+0E04</td>
<td>ฅ</td>
<td>Thai</td>
<td>THAI CHARACTER KHO KHWAII</td>
<td>Thai,und-Thai-cons</td>
<td></td>
<td>[5], [100], [101]</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Variant Sets

Summary

<table>
<thead>
<tr>
<th>Number of variant sets</th>
<th>30</th>
</tr>
</thead>
<tbody>
<tr>
<td>Largest variant set</td>
<td>4</td>
</tr>
<tr>
<td>Ordinary Variants by Type</td>
<td>blocked (98)</td>
</tr>
</tbody>
</table>

The following tables list all variant sets defined in this LGR, except for singleton sets. Each table lists all variant mapping pairs of the set; one per row. Mappings are assumed to be symmetric: each row documents both forward (→) and reverse (←) mapping directions. In each table, the mappings are sorted by Source value in ascending code point order; shading is used to group mappings from the same source code point or sequence.

Where the type of both forward and reverse mappings are the same, a single value is given in the Type(s) column, otherwise the types for forward and reverse mappings, as well as comments and references are listed above one another.

A mapping where source and target are the same is reflexive. Variant sets consisting of only a single reflexive mapping are not shown as a set. Instead, the variant type of the mapping is listed in the Variants column of the Repertoire by Code Point table. Reflexive mappings that are part of a larger set are indicated with a “*”.

In any LGR with variant specifications that are well behaved, all members within each variant set are defined as variants of each other; the mappings in each set are symmetric and transitive; and all variant sets are disjoint.

Common Legend

Source: Source of the mapping pair.
Target: Destination of the mapping pair.
Glyph: The shape displayed for source or target depends on the fonts available to your browser.
→ - forward: Indicates that variant Type, References and Comment apply to the mapping from source to target.
← - reverse: Indicates that variant Type, References and Comment apply to the reverse mapping from target to source.
↔ - both: Indicates that variant Type, References and Comment apply to both forward and reverse mapping.
= - reflexive: Indicates that variant Type, References and Comment are for a reflexive mapping where source equals target.
☐ - not in LGR: Indicates that variant is not in LGR.

Type: The type of the variant mapping. There are some predefined variant types such as "allocatable" and "blocked", while others are defined specifically for each LGR.
References: One or more reference IDs (optional). A "*" separates references for reverse / forward mappings, if different.
Comment: A descriptive comment (optional). A "*" separates comments for reverse / forward mappings, if different.

Variant Set 1 — 3 Members - 3 Mappings

<table>
<thead>
<tr>
<th>#</th>
<th>Source</th>
<th>Glyph</th>
<th>Target</th>
<th>Glyph</th>
<th>Type(s)</th>
<th>References</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>U+1200</td>
<td>U</td>
<td>U+1210</td>
<td></td>
<td>↔ blocked</td>
<td>[4]</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>U+1200</td>
<td>U</td>
<td>U+1280</td>
<td></td>
<td>↔ blocked</td>
<td>[4]</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>U+1210</td>
<td></td>
<td>U+1280</td>
<td></td>
<td>↔ blocked</td>
<td>[4]</td>
<td></td>
</tr>
</tbody>
</table>
## Classes, Rules and Actions

### Character Classes

The following table lists all top-level classes with their definition and the regular expression defining their members.

<table>
<thead>
<tr>
<th>Name</th>
<th>Definition</th>
<th>Members</th>
<th>References</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>und-Laoo-Cf</td>
<td>Tag= und-Laoo-Cf</td>
<td>{U+0E81 U+0E87 U+0E8A U+0E8D U+0E94 U+0E97 U+0E99 U+0E9A U+0E9F U+0EA1 U+0EA3 U+0EA5 U+0EA7 U+0EAA}</td>
<td></td>
<td></td>
</tr>
<tr>
<td>und-Laoo-consonant</td>
<td>Tag= und-Laoo-consonant</td>
<td>{U+0E81 U+0E82 U+0E84 U+0E87 U+0E88 U+0E8A U+0E8D U+0E94 U+0E95 U+0E96 U+0E97 U+0E99 U+0E9A U+0EBB U+0E9C ...}</td>
<td></td>
<td></td>
</tr>
<tr>
<td>und-Laoo-semi-consonant</td>
<td>Tag= und-Laoo-semi-consonant</td>
<td>{U+0E0C}</td>
<td></td>
<td></td>
</tr>
<tr>
<td>und-Laoo-tone-mark</td>
<td>Tag= und-Laoo-tone-mark</td>
<td>{U+0EC8 U+0EC9 U+0ECA U+0ECB}</td>
<td></td>
<td></td>
</tr>
<tr>
<td>und-Laoo-vowel-above</td>
<td>Tag= und-Laoo-vowel-above</td>
<td>{U+0EB1 U+0EB4 U+0EB5 U+0EB6 U+0EB7 U+0EBB U+0ECD}</td>
<td></td>
<td></td>
</tr>
<tr>
<td>und-Laoo-vowel-below</td>
<td>Tag= und-Laoo-vowel-below</td>
<td>{U+0EB8 U+0EB9}</td>
<td></td>
<td></td>
</tr>
<tr>
<td>und-Thai-av</td>
<td>Tag= und-Thai-av</td>
<td>{U+0E31 U+0E34 U+0E35 U+0E36 U+0E37}</td>
<td></td>
<td>Any above vowel</td>
</tr>
<tr>
<td>und-Thai-below-av</td>
<td>Tag= und-Thai-below-av</td>
<td>{U+0E38 U+0E39}</td>
<td></td>
<td>Any below vowel</td>
</tr>
<tr>
<td>und-Thai-c-av-bv</td>
<td>([und-Thai-consonant:][und-Thai-above-vowel:][und-Thai-below-vowel:])</td>
<td>{U+0E01 U+0E02 U+0E03 U+0E04 U+0E05 U+0E06 U+0E07 U+0E08 U+0E09 U+0E0A U+0E0B U+0E0C U+0E0D U+0E0E U+0E0F ...}</td>
<td></td>
<td>Any consonant, vowel-above or vowel-below</td>
</tr>
<tr>
<td>und-Thai-cons</td>
<td>Tag= und-Thai-cons</td>
<td>{U+0E01 U+0E02 U+0E03 U+0E04 U+0E05 U+0E06 U+0E07 U+0E08 U+0E09 U+0E0A U+0E0B U+0E0C U+0E0D U+0E0E U+0E0F ...}</td>
<td></td>
<td>Any Consonant</td>
</tr>
<tr>
<td>und-Thai-ct</td>
<td>([und-Thai-consonant:][und-Thai-tone:])</td>
<td>{U+0E01 U+0E02 U+0E03 U+0E04 U+0E05 U+0E06 U+0E07 U+0E08 U+0E09 U+0E0A U+0E0B U+0E0C U+0E0D U+0E0E U+0E0F ...}</td>
<td></td>
<td>Any consonant or tone</td>
</tr>
<tr>
<td>und-Thai-ctaa</td>
<td>([und-Thai-consonant:][und-Thai-tone:][und-Thai-sara-aa:])</td>
<td>{U+0E01 U+0E02 U+0E03 U+0E04 U+0E05 U+0E06 U+0E07 U+0E08 U+0E09 U+0E0A U+0E0B U+0E0C U+0E0D U+0E0E U+0E0F ...}</td>
<td></td>
<td>Any consonant, tone or sara-aa</td>
</tr>
<tr>
<td>und-Thai-sara-aa</td>
<td>Tag= und-Thai-sara-aa</td>
<td>{U+0E32}</td>
<td></td>
<td>SARA AA</td>
</tr>
<tr>
<td>und-Thai-tone</td>
<td>Tag= und-Thai-tone</td>
<td>{U+0E48 U+0E49 U+0E4A U+0E4B}</td>
<td></td>
<td>Any tone mark</td>
</tr>
<tr>
<td></td>
<td></td>
<td>{U+1E90 U+1E91 U+1E92 U+1E93 U+1E94 U+1E95 U+1E96 U+1E97}</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### Whole label evaluation and context rules in LGR

The following table lists all the top-level, or named rules defined in the LGR and indicates whether they are used as trigger in an action or as context (when or not-when) for a code point. (Any use of context rules for variants is not indicated.)

<table>
<thead>
<tr>
<th>Name</th>
<th>Regular Expression</th>
<th>Used as Trigger</th>
<th>Used as Context</th>
<th>Anchor</th>
<th>References</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Common-leading-combining-mark</td>
<td>(start) ([[:class property:gc=Mn:]u[:class property:gc=Mc:]])</td>
<td>True</td>
<td>False</td>
<td>False</td>
<td>None</td>
<td>None</td>
</tr>
<tr>
<td>und-Laoo-follows-consonant</td>
<td>([[:und-Laoo-consonant:]])</td>
<td>False</td>
<td>True</td>
<td>True</td>
<td>WLE Rule No. 1; semi-consonant must follow a consonant</td>
<td></td>
</tr>
<tr>
<td>und-Laoo-precedes-consonant</td>
<td>([[:und-Laoo-consonant:]] →)</td>
<td>False</td>
<td>True</td>
<td>True</td>
<td>WLE Rule No. 2; vowel-before precedes a main consonant cluster</td>
<td></td>
</tr>
<tr>
<td>und-Laoo-follows-main-consonant</td>
<td>([[:und-Laoo-consonant:]][[:und-Laoo-semi-consonant:]]) →</td>
<td>False</td>
<td>True</td>
<td>True</td>
<td>WLE Rule No. 3; vowel-above, and vowel-below follow a main consonant C</td>
<td></td>
</tr>
<tr>
<td>und-Laoo-follows-C-tonemark-vabovelow</td>
<td>([[:und-Laoo-consonant:]][[:und-Laoo-semi-consonant:]][[:und-Laoo-tone-mark:][[:und-Laoo-vowel-above:]]] →</td>
<td>False</td>
<td>True</td>
<td>True</td>
<td>WLE Rule No. 4; vowel-after follows a main consonant, tone-mark or vowel-above</td>
<td></td>
</tr>
<tr>
<td>und-Laoo-consonant-cluster</td>
<td>([[:und-Laoo-consonant:];{1,2}[:und-Laoo-semi-consonant:];{0,1}]</td>
<td>False</td>
<td>False</td>
<td>False</td>
<td>Defining consonant cluster for Rule No. 5</td>
<td>WLE Rule No. 5; The sequence (0EB2 0EB0) follows a vowel before, and a consonant cluster</td>
</tr>
<tr>
<td>und-Laoo-follows-vbefore-consonant-cluster</td>
<td>(U+0EC0([[:und-Laoo-consonant-cluster:]]) →</td>
<td>False</td>
<td>True</td>
<td>True</td>
<td>WLE Rule No. 6; A tone-mark follows a main consonant, vowel-above or vowel-below</td>
<td></td>
</tr>
<tr>
<td>und-Laoo-follows-C-vabove-vbelow</td>
<td>([[:und-Laoo-consonant:]][[:und-Laoo-semi-consonant:]][[:und-Laoo-vowel-above:]][[:und-Laoo-vowel-below:]]) →</td>
<td>False</td>
<td>True</td>
<td>True</td>
<td>WLE Rule No. 7; The sign 0ECC can only occur after final consonants</td>
<td></td>
</tr>
<tr>
<td>und-Laoo-follows-Cf</td>
<td>([[:und-Laoo-Cf:]]) →</td>
<td>False</td>
<td>True</td>
<td>True</td>
<td>WLE Rule No. 8; The sign 0EC6 can only occur 0 to 3 times at the end of the label</td>
<td></td>
</tr>
<tr>
<td>und-Laoo-repetition-mark-limit</td>
<td>([[:und-Laoo-consonant:]] → (U+0EC6){2,0}(end))</td>
<td>False</td>
<td>True</td>
<td>True</td>
<td>WLE 7.2: check if current cp is preceding a consonant</td>
<td></td>
</tr>
<tr>
<td>und-Thai-precedes-consonant</td>
<td>([[:und-Thai-consonant:]])</td>
<td>False</td>
<td>True</td>
<td>True</td>
<td>WLE 7.3: check if current cp is following a consonant</td>
<td></td>
</tr>
</tbody>
</table>
## Actions

The following table lists the actions that are used to assign dispositions to labels and variant labels, based on the specified conditions. The order of actions defines their precedence: the first action triggered by a label is the one defining its disposition.

<table>
<thead>
<tr>
<th>#</th>
<th>Condition</th>
<th>Rule / Variant Set</th>
<th>Disposition</th>
<th>References</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>if label match</td>
<td>Common-leading-combining-mark</td>
<td>invalid</td>
<td></td>
<td>any variant label with a code point out of repertoire is invalid</td>
</tr>
<tr>
<td>2</td>
<td>if at least one variant is in {out-of-repertoire-var}</td>
<td>invalid</td>
<td></td>
<td></td>
<td>labels must not commence with a combining mark ´</td>
</tr>
<tr>
<td>3</td>
<td>if label match</td>
<td>Common-leading-combining-mark</td>
<td>invalid</td>
<td></td>
<td>any variant label with a code point out of repertoire is invalid</td>
</tr>
<tr>
<td>4</td>
<td>if at least one variant is in {out-of-repertoire-var}</td>
<td>invalid</td>
<td></td>
<td></td>
<td>any variant label with a code point out of repertoire is invalid</td>
</tr>
<tr>
<td>5</td>
<td>if label match</td>
<td>Common-leading-combining-mark</td>
<td>invalid</td>
<td></td>
<td>any variant label with a code point out of repertoire is invalid</td>
</tr>
<tr>
<td>6</td>
<td>if at least one variant is in {out-of-repertoire-var}</td>
<td>invalid</td>
<td></td>
<td></td>
<td>any variant label with a code point out of repertoire is invalid</td>
</tr>
</tbody>
</table>

**Legend**

(...)-variant type set: In the "Rule/Variant Set" column the notation {...} means a set of variant types.

### Table of References

- [0] The Unicode Standard 1.1, The Unicode Consortium, Mountain View, CA. 1993
  
  *Ethiopic Script Versions for the Eight Languages*

- [2] Corpus Analysis performed by crawling 598 html and 40 PDF files with Tigrigna Contents published online, August 2016
  
  *Cited as Auxiliary Evidence for Tigrigna Code Points*

- [3] Corpus Analysis performed by crawling 14,850 html Amharic Contents of size 1.8 GB published online, August 2016
  
  *Cited as Auxiliary Evidence for Amharic Code Points*

  
  *Cited for Amharic-Driven Variants in Ethiopic Script*

- [5] The Unicode Standard 1.1
Table of References

[0] The Unicode Standard 1.1, The Unicode Consortium, Mountain View, CA. 1993


Ethiopic Script Versions for the Eight Languages

[2] Corpus Analysis performed by crawling 598 html and 40 PDF files with Tigrigna Contents published online, August 2016
Cited as Auxiliary Evidence for Tigrigna Code Points

[3] Corpus Analysis performed by crawling 14,850 html Amharic Contents of size 1.8 GB published online, August 2016
Cited as Auxiliary Evidence for Amharic Code Points

Cited for Amharic-Driven Variants in Ethiopic Script

[5] The Unicode Standard 1.1


[201] Lao grammar book published by the Ministry of Education in 1967, see Appendix B, Figure 1

[202] Lao grammar book published by the Ministry of Education in 1967, see Appendix B, Figure 2

[203] Lao grammar book published by the Ministry of Education in 1967, see Appendix B, Figure 3
Validate a Label
1. Click on “Import” to load single or multiple existing LGR files. See “Import” for details.

2. Always view “Validate LGR” output as the tool checks the loaded LGR(s) during this process.

3. For validating a label, click on “Validate label” button.
Validate a Label with a Single LGR

1. Enter the label to be validated

2. Click on the “Validate” button

3. Check to include mixed-script variants
Validate a Label with a Single LGR

Result given by “Validate label” shows whether the label is valid or invalid and also lists its variants, their dispositions and the rules/actions against which the label or any of its variants is valid or invalid.
1. Enter the label to be validated

2. Select a script from the list of the scripts of different LGRs forming the merged LGR

3. Click on the “Validate” button
Validate a Label with a Merged LGR

Result given by “Validate label” shows whether the label is valid or invalid and also lists its variants, their dispositions and the rules/actions of which the label or any of its variants is valid or invalid.
Review Impact on Existing Labels by Revising an Existing LGR
Click on “Diff labels of two LGRs” button to determine differences caused by modifying an LGR.
Enter Details

1. Select first LGR
2. Select second LGR
3. Select file containing labels
4. Check collisions if you want to check label collisions as well
5. Check "Output rules" if you want to check output rules for each label
6. Click on "Get diff" button
Download Results from Task Status Page

1. Click on the download link on the homepage to get the “Diff labels of two LGRs” results
Download Results

1. Click on Home link for LGR Tool to get to this page

2. Click on the download link on the homepage to get the “Diff labels of two LGRs” results
Results

# Labels not in LGR 1 #

# Labels not in LGR 2 #

Label نادي

# LGR comparison #

Label نادي not in LGR proposed-arabic-lgr-18092017-en

## Comparison on label 'کلکت' [U+06A9 U+0644 U+0643 U+062A U+06C1]

### Test dispositions: ###

```
No changes in disposition.
```

### Test number of variants: ###

```
No changes in number of variants.
```

## Comparison on label 'نبأ' [U+0644 U+0642 U+0627 U+0621]

### Test dispositions: ###

```
No changes in disposition.
```

### Test number of variants: ###

```
No changes in number of variants.
```

Specifies changes in disposition of labels by the revised LGR

Specifies new variants of labels formed by the revised LGR
Review Impact on Existing Labels by Introducing a New LGR
Select Get Collisions in a List of Labels

Click on “Get collisions in a list of labels” button to determine label collisions from an existing file if a new LGR is introduced – for example, two unique labels become variants of each other.
Enter Details

1. Select LGR
2. Select Labels file
3. Check with existing TLDs
4. Check “Full Dump” to get summary of each operation done on the labels
6. Click on “Get collisions”
Download Results from Task Status Page

<table>
<thead>
<tr>
<th>Tasks</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Review of 1 IDN table - Oct. 20, 2022, 10:35 a.m.</td>
<td>Download report</td>
</tr>
<tr>
<td>Annotate labels on LGR 5 - Oct. 22, 2022, 5:12 p.m.</td>
<td>Download report</td>
</tr>
<tr>
<td>Diff with LGR Demo-1 - Oct. 22, 2022, 5:46 p.m.</td>
<td>Download report</td>
</tr>
<tr>
<td>Collision with LGR lgr-second-level-arabic-language-31may22-en - Oct. 22, 2022, 5:52 p.m.</td>
<td>Download report</td>
</tr>
</tbody>
</table>

Click on the download link to get the “Collisions” results
Download Results

Click on Home link for LGR Tool to get to this page

Advanced LGR (Label Generation Ruleset) Tools

This application provides a convenient interface for browsing and editing LGRs conforming to the Representing Label Generation Rulesets using XML specification.

Previously loaded LGR file(s)
Previously, you edited the following LGR file(s). Click on its title to resume your editing session.

LGRs
- View lgr-new
- View sample-french
- View union-of-sample-french-and-lgr-4-arabic-script-29jun20-en

Create a new LGR file or import an existing one

Import an existing XML file

Note that importing large LGR files may take significant time to load on your browser.

Start with a New blank XML file

Start from a built-in LGR

The following LGRs are pre-installed in the system. You may use them as a starting point for your own LGR. To do so,

Your saved results
The following files contain your tool computation results.

⚠️ Note that these files could be cleaned up regularly.
- Download 20210528_102129_collisions_lgr-4-arabic-script-29jun20-en.txt.gz

Click on the download link to get the “Collisions” results
Results

# Labels not in LGR #

Label: 'مكتبة' | 'مكتبة'
Code points: [U+06AA U+0644 U+0643 U+062A U+06C1] | [U+06AA U+0644 U+0643 U+062A U+06C1]
Category: Primary | Primary

# Collisions #

## Collision ##

```
Label: 'مكتبة' | 'مكتبة'
Code points: [U+06AA U+0644 U+0643 U+062A U+06C1] | [U+06AA U+0644 U+0643 U+062A U+06C1]
Category: Primary | Primary
```

## Details for label 'مكتبة' [U+06AA U+0644 U+0643 U+062A U+06C1] ##

```
Variant 'مكتبة' [U+06AA U+0644 U+0643 U+062A U+0629]:
  Disposition: invalid

Rules:
```

## Details for label 'مكتبة' [U+06AA U+0644 U+0643 U+062A U+06C1] ##

```
Variant 'مكتبة' [U+06AA U+0644 U+0643 U+062A U+0629]:
  Disposition: invalid
```

List of labels which have become invalid by introducing a new LGR

Details of a label and its corresponding variant(s)
Validate Multiple Labels
Using a Single or Merged LGR(s)
Select Generate Disposition Annotations

1. Click on “Import” to load existing LGR file(s). See “Import” for details.

2. Click on “Generate disp. annotations” button to validate labels given in a text file.
Enter Details for Single LGR

1. Select LGR
2. Select Labels file for validation. The labels file contains single label per line in UTF-8 format
3. Click on “Annotate” button
Enter Details for Merged LGR

1. Select LGR
   Optional file of existing labels to check for collisions

2. Select a script from the list of the scripts of different LGRs forming the merged LGR

3. Select Labels file for validation

4. Click on “Annotate” button
Download Results from Task Status Page

Tasks

Tasks are ordered from newest to latest

<table>
<thead>
<tr>
<th>Task Description</th>
<th>Date/Time</th>
<th>Download link</th>
<th>Status</th>
<th>Report Expiry</th>
</tr>
</thead>
<tbody>
<tr>
<td>Annotate labels on LGR 5</td>
<td>July 25, 2023, 5:05 p.m.</td>
<td>Download report</td>
<td>Success</td>
<td>15 days</td>
</tr>
<tr>
<td>Review of 1 IDN table</td>
<td>July 21, 2023, 6:57 a.m.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Review of 1 IDN table</td>
<td>July 13, 2023, 2:56 p.m.</td>
<td>Download report</td>
<td>Success</td>
<td>3 days</td>
</tr>
<tr>
<td>Review of 1 IDN table</td>
<td>July 13, 2023, 2:55 p.m.</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

When you delete a task, the corresponding report is not deleted

Delete completed tasks

Click on the download link to get the “Generate disp. annotations” results
Advanced LGR (Label Generation Ruleset) Tools

This application provides a convenient interface for browsing and editing LGRs conforming to the Representing Label Generation Rulesets using XML specification.

Previously loaded LGR file(s)
Previously, you edited the following LGR file(s). Click on its title to resume your editing session.
LGRs
- View lgr-new
- View sample-french
- View union-of-sample-french-and-lgr-4-arabic-script-29jun20-en

Your saved results
The following files contains your tools computation results.
⚠️ Note that these files could be cleaned up regularly.
- Download 20210528_102129_collisions_lgr-4-arabic-script-29jun20-en.txt.gz

Create a new LGR file or import an existing one
Import an existing XML file
⚠️ Note that importing large LGR files may take significant time to load on your browser.
Start with a New blank XML file
Start from a built-in LGR
The following LGRs are pre-installed in the system. You may use them as a starting point for your own LGR. To do so,
Results

Validation result of each label is written next to it. For invalid labels, the tool also specifies the reason/rule due to which the label is invalid.

- Code point U+0EB3 not in repertoire
- Code point U+0EC2 does not comply with rules 'precedes-consonant'
- Code point U+0EDD not in repertoire
- Code point U+0EC9 does not comply with rules 'follows-C-vabove-vbelow'
View Cross-Script Variants of Labels
Select Cross-script Variants

1. Click on “Import” to load single or multiple LGR file(s). See “Import” for details.

2. Click on “Cross-script variants” button to view cross-script variants of labels given in a text file.
Enter Details

1. Select single LGR or a merged LGR
2. Select a script from the list of the scripts of different LGRs forming the merged LGR
3. Select Labels file for validation
4. Click on “Launch” button
Download Results from Task Status Page

<table>
<thead>
<tr>
<th>Tasks</th>
<th>Download report</th>
<th>Status</th>
<th>Delete</th>
</tr>
</thead>
<tbody>
<tr>
<td>Review of 1 IDN table - Oct. 20, 2022, 10:35 a.m.</td>
<td>✔️</td>
<td>Success</td>
<td>✔️</td>
</tr>
<tr>
<td>Annotate labels on LGR 5 - Oct. 22, 2022, 5:12 p.m.</td>
<td>✔️</td>
<td>Success</td>
<td>✔️</td>
</tr>
<tr>
<td>Diff with LGR Demo-1 - Oct. 22, 2022, 5:46 p.m.</td>
<td>✔️</td>
<td>Success</td>
<td>✔️</td>
</tr>
<tr>
<td>Collision with LGR Igr-second-level-arabic-language-31may22-en - Oct. 22, 2022, 5:52 p.m.</td>
<td>✔️</td>
<td>Success</td>
<td>✔️</td>
</tr>
</tbody>
</table>

Click on the download link to get the “Cross-script variants” results.
Advanced LGR (Label Generation Ruleset) Tools

This application provides a convenient interface for browsing and editing LGRs conforming to the Representing Label Generation Rulesets using XML specification.

Previously loaded LGR file(s)
Previously, you edited the following LGR file(s). Click on its title to resume your editing session.

LGRs
- View lgr-new
- View sample-french
- View union-of-sample-french-and-lgr-4-arabic-script-29jun20-en

Create a new LGR file or import an existing one

[Import an existing XML file]

[Start with a New blank XML file]

Start from a built-in LGR
The following LGRs are pre-installed in the system. You may use them as a starting point for your own LGR. To do so:

Your saved results
The following files contains your tools computation results.

⚠️ Note that these files could be cleaned up regularly.

- Download 20210528_102129_collisions_lgr-4-arabic-script-29jun20-en.txt.gz

Click on the download link to get the “Cross-script variants” results
Results

Cross-script variants of the label

Disposition of cross-script variants

Specifies the LGR (from a set of multiple LGRs) that is used to generate these cross-script variants

Input label U+0561 U+0575 U+0580 (ωη) has cross-script variants:
- Cross-variant U+0448 U+0575 U+0580 (ωη), disposition invalid:
  + U+0448 (ω): Cyril
Input label U+0581 U+0561 U+0579 U+0586 (γως) has cross-script variants:
- Cross-variant U+0581 U+0448 U+0579 U+0586 (γως), disposition invalid:
  + U+0448 (ω): Cyril
  + Cross-variant U+0067 U+0561 U+0579 U+0586 (γως), disposition invalid:
    + U+0067 (g): Latn
- Cross-variant U+0067 U+0448 U+0579 U+0586 (γως), disposition invalid:
  + U+0448 (ω): Cyril
  + U+0067 (g): Latn
Input label U+0566 U+0578 U+057D (qνω) has cross-script variants:
- Cross-variant U+0566 U+0578 U+0075 (qνω), disposition invalid:
  + U+0075 (ω): Latn
  + Cross-variant U+0566 U+006E U+057D (qνω), disposition invalid:
    + U+006E (n): Latn
- Cross-variant U+0566 U+006E U+0075 (qνω), disposition invalid:
  + U+0075 (ω): Latn
  + U+006E (n): Latn
- Cross-variant U+0071 U+0578 U+057D (qνω), disposition invalid:
  + U+0071 (q): Latn
- Cross-variant U+0071 U+0578 U+0075 (qνω), disposition invalid:
  + U+0071 (q): Latn
  + U+0075 (ω): Latn
- Cross-variant U+0071 U+006E U+057D (qνω), disposition invalid:
  + U+0071 (q): Latn
  + U+006E (n): Latn
- Cross-variant U+0071 U+006E U+0075 (qνω), disposition invalid:
  + U+0071 (q): Latn
  + U+0075 (ω): Latn
  + U+006E (n): Latn
Compare LGRs
Compare LGRs

Click on “Compare two LGRs” under the “Tools” tab for comparing LGRs.
Union, Intersection or Difference

1. Select first LGR
2. Select second LGR
3. Select “Union”, “Intersection” or “Diff” to perform relevant function on the two LGRs
4. Click on the “Compare” button
With Merged LGRs, Only Difference

1. Select first LGR

2. Select second LGR

“Diff” selected by default

3. Click on the “Compare” button

**Note:**
Comparison will be performed only between two LGR sets or two simple LGRs, union and intersection are not available for LGR sets.
Result of Difference function

Important Note: These operations only provide provisional results which must be manually reviewed and finalized.
Harmonize Multiple LGRs
Select Harmonization

1. Click on “Import” to load single or multiple LGR file(s). See “Import” for details

2. Click on “Harmonization” button to harmonize two LGR files
Enter Details

1. Select first LGR

2. Select second LGR

Optional RootZone LGR to infer new variant sets

3. Click on “Harmonize” button
Results of harmonization for Sample-French and Greek Testing txt LGR

Resulting LGRs

- View harmonized version of Sample-French_harmonized_20230808_151748
- View harmonized version of Greek Testing txt LGR_harmonized_20230808_151748

The Harmonized LGRs will also be listed on the homepage

If you have questions, please contact globalsupport@icann.org
Compute Variants
Compute Variants

1. Click on “Import” to load single or multiple LGR file(s). See “Import” for details

2. Click on “Compute Variants” to compute variants of label(s) against imported LGR
Enter Details

1. Imported LGR
2. File which contains labels to be used for variant computation
3. Press this to compute variants
## Results

### Tasks

Tasks are ordered from newest to latest

<table>
<thead>
<tr>
<th>Task Description</th>
<th>Date/Time</th>
<th>Status</th>
<th>Report Expiry</th>
</tr>
</thead>
<tbody>
<tr>
<td>Variant computation on LGR lgr-second-level-arabic-script-31may22-en (1)</td>
<td>Aug. 2, 2023, 12:54 p.m.</td>
<td>Success</td>
<td>15 days</td>
</tr>
<tr>
<td>Review of 1 IDN table</td>
<td>July 31, 2023, 12:43 p.m.</td>
<td>Success</td>
<td>13 days</td>
</tr>
<tr>
<td>Annotate labels on LGR 5</td>
<td>July 25, 2023, 5:05 p.m.</td>
<td>Success</td>
<td>8 days</td>
</tr>
<tr>
<td>Review of 1 IDN table</td>
<td>July 21, 2023, 6:57 a.m.</td>
<td>Success</td>
<td>3 days</td>
</tr>
</tbody>
</table>

When you delete a task, the corresponding report is not deleted.

**Delete completed tasks**

---

**Download results of compute variants**
Welcome to the LGR (Label Generation Ruleset) Tool

This application provides a convenient interface for browsing and editing LGRs conforming to the Representing Label Generation Rulesets using XML specification.

Previously loaded LGR file(s)
Previously, you edited the following LGR file(s). Click on its title to resume your editing session.
LGRs
- View lgr-4-arabic-script-29jun20-en
- View lgr-4-devanagari-script-29jun20-en

Create a new LGR file or import an existing one

Note that importing large LGR files may take significant time to load on your browser.

Your saved results
The following files contains your tools computation results.

Note that these files could be cleaned up regularly.
- Download 20201120_092126_labels_variants_lgr-4-devanagari-script-29jun20-en.csv.gz

Press here to go to homepage. The results will be present on the Homepage when available

Download results of compute variants
Label Forms

A label, U label against IDNA2008
Label Forms

- Type in a label or upload a list of labels

Welcome to the LGR (Label Generation Ruleset) Tools

Select your mode:
- Validate label(s) against an LGR
- Review IDN table(s)

This function is available for all modes in the LGR tool. The purpose of this is to provide the U-label/ A-label and codepoints of the input label(s).
Label Forms

Display all label forms

Check labels one by one

Label

Label can be in U-Label or A-Label form or a list of code points.

Click to see the result label form

Display forms

Get label forms on a list of labels

Labels
Choose file
No file chosen

Download forms

File should be a text file encoded in UTF-8 and using 0x0A line ending. It must contain one label per line in U-Label or A-Label form or as a list of code points. Comments begin with '#'
Label Forms - Input One Label (Example 1 - valid)

Display all label forms

1. Input is Unicode label

2. Click to see the result

<table>
<thead>
<tr>
<th>Code point sequence</th>
<th>U-label</th>
<th>A-label</th>
</tr>
</thead>
<tbody>
<tr>
<td>U+674E U+5B50</td>
<td>李子</td>
<td>xn--i8sx2z</td>
</tr>
</tbody>
</table>

Code point sequence, U-label, A-label - all 3 forms will be generated based on the input.
Label Forms - Input One Label (Example 1 - invalid)

Display all label forms

1. Input is Unicode label

2. Click to see the result

Label can be in U-Label or A-Label form or a list of code points.

<table>
<thead>
<tr>
<th>Code point sequence</th>
<th>U-label</th>
<th>A-label</th>
</tr>
</thead>
<tbody>
<tr>
<td>U+674E U+5B50 U+3002</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

Warning Note

李子。 is invalid as it contains full stop (dot).

Code point sequence will be generated, U-label, A-label forms will not be generated.
## Label Forms - Input One Label (Example 2 - valid)

1. **Input is Code point sequence**

   - **Label**: 
     - U+101D U+1031 U+1006 U+102C

2. **Click to see the result**

   - **Code point sequence**: U+101D U+1031 U+1006 U+102C
   - **U-label**: ᶨᵒʳפוס
   - **A-label**: xn--tid9b4bs

---

**Code point sequence, U-label, A-label**
- All 3 forms will be generated based on the input
Label Forms - Input One Label (Example 2 - invalid)

Display all label forms

Warning Note

1. Input is Unicode label

Label

<table>
<thead>
<tr>
<th>Code point sequence</th>
<th>U-label</th>
<th>A-label</th>
</tr>
</thead>
<tbody>
<tr>
<td>U+101D U+1031 U+1006 U+102C U+3002</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

117

is invalid as it contains full stop (dot).

2. Click to see the result

Code point sequence will be generated, U-label, A-label forms will not be generated.
Label Forms - Input a List of Labels (Example 3)

Upload file to check the list of labels (in txt format)

Get label forms on a list of labels

Download the result file of Label forms (in csv format)
Label Forms - Input a List of Labels (Example 3 - Input)

- Input file format is .txt.
- Input can be one label per line.
- Each label can be in:
  - Code point sequence
  - Unicode
  - Punycode
- The lines beginning with ‘#’ are comments and will not be processed.
### Label Forms - Input a List of Labels (Example 3 - Output)

<table>
<thead>
<tr>
<th>Input</th>
<th>Code point sequence</th>
<th>U-label</th>
<th>A-label</th>
<th>Note</th>
</tr>
</thead>
<tbody>
<tr>
<td>U+1005 U+102D U+1014 U+103A</td>
<td>U+1005 U+102D U+1014 U+103A</td>
<td>⚗️</td>
<td>xn--sid4a9d7b</td>
<td>-</td>
</tr>
<tr>
<td>U+1005 U+102D U+1014 U+103A U+002D</td>
<td>U+1005 U+102D U+1014 U+103A U+002D</td>
<td>-</td>
<td>-</td>
<td>⚗️- is invalid due to hyphen restrictions in the RFC5891 as it ends with a hyphen-minus.</td>
</tr>
<tr>
<td>⚴</td>
<td>U+101B U+1010 U+1014 U+102C</td>
<td>⚴</td>
<td>xn--3idit4e</td>
<td>-</td>
</tr>
<tr>
<td>⚴</td>
<td>U+101B U+1010 U+002D U+002D U+1014 U+102C</td>
<td>-</td>
<td>-</td>
<td>⚴- is invalid due to hyphen restrictions in the RFC5891 as it contains hyphen-minus in the third and fourth positions.</td>
</tr>
<tr>
<td>xn--ridd1ji0d</td>
<td>U+1006 U+102D U+102F U+1004 U+103A</td>
<td>⚴</td>
<td>xn--ridd1ji0d</td>
<td>-</td>
</tr>
<tr>
<td>xn--vi8hua2f</td>
<td>U+D83C U+DF55 U+D83C U+DF5F U+D83C U+DF79</td>
<td>-</td>
<td>-</td>
<td>'utf-8' codec can't encode characters in position 55-60: surrogates not allowed</td>
</tr>
</tbody>
</table>

- **Output csv:** *U-label* and *A-label* forms will be generated when
  - Input is Code point sequence and valid
  - Input is U-label and valid
  - Input is A-label and valid
### Label Forms - Input a List of Labels (Example 3 - Output)

<table>
<thead>
<tr>
<th>Input</th>
<th>Code point sequence</th>
<th>U-label</th>
<th>A-label</th>
<th>Note</th>
</tr>
</thead>
<tbody>
<tr>
<td>U+1005 U+102D U+1014 U+103A</td>
<td>U+1005 U+102D U+1014 U+103A</td>
<td>❗️</td>
<td>xn–sid4a9d7b</td>
<td>-</td>
</tr>
<tr>
<td>U+1005 U+102D U+1014 U+103A U+002D</td>
<td>U+1005 U+102D U+1014 U+103A U+002D</td>
<td>-</td>
<td>-</td>
<td>❗️ is invalid due to hyphen restrictions in the RFC5891 as it ends with a hyphen-minus.</td>
</tr>
<tr>
<td>ᶜᵒᵒʳᵉ</td>
<td>U+101B U+1010 U+1014 U+102C</td>
<td>ᶜᵒᵒʳᵉ</td>
<td>xn–3idit4e</td>
<td>-</td>
</tr>
<tr>
<td>ᶜᵒᵒʳ―ᵣ</td>
<td>U+101B U+1010 U+002D U+002D U+1014 U+102C</td>
<td>-</td>
<td>-</td>
<td>❗️ is invalid due to hyphen restrictions in the RFC5891 as it contains hyphen-minus in the third and fourth positions.</td>
</tr>
<tr>
<td>xn–ridd1ji0d</td>
<td>U+1006 U+102D U+102F U+1004 U+103A</td>
<td>❗️</td>
<td>xn–ridd1ji0d</td>
<td>-</td>
</tr>
<tr>
<td>xn–vi8hua2f</td>
<td>U+D83C U+DF55 U+D83C U+DF5F U+D83C U+DF79</td>
<td>-</td>
<td>-</td>
<td>'utf-8' codec can't encode characters in position 55-60: surrogates not allowed</td>
</tr>
</tbody>
</table>

- **Output csv**: Notes will be generated, (with no A-label or U-label) when
  - Input is Code point sequence and IDNA2008-non-compliant
  - Input is U-label and IDNA2008-non-compliant
  - Input is A-label and IDNA2008-non-compliant
Engage with ICANN and IDN Program

Thank You and Questions
Reach us at: IDNProgram@icann.org
Website: icann.org/idn

twitter.com/icann
gplus.to/icann
facebook.com/icannorg
weibo.com/ICANNorg
linkedin.com/company/icann
flickr.com/photos/icann
youtube.com/user/icannnews
slideshare.net/icannpresentations