Introduction to LGR Toolset

- 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. Validate label(s) against an LGR
   (pp. 13-19)

3. Advanced LGR Tools
   (pp. 20-21)

4. Importing an existing LGR in a toolset
   (pp. 22-25)

5. Importing multiple LGRs in a toolset
   (pp. 26-29)

6. Creating an LGR via toolset
   (pp. 30-56)
Agenda

7. Validating LGR (pp. 57-60)

8. Viewing LGR as XML and HTML (pp. 61-72)

9. Validating a label using an LGR (pp. 73-78)

10. Reviewing impact on existing labels by revising an existing LGR (pp. 79-84)

11. Reviewing impact on existing labels by introducing a new LGR (pp. 85-90)

12. Validating multiple labels using LGR(s) (pp. 91-97)
Agenda (cont.)

13. Viewing cross-script variants of labels (pp. 98-103)

14. Comparing LGRs (pp. 104-108)

15. Harmonizing multiple LGRs (pp. 109-112)

16. Computing variants (pp. 113-116)
Agenda Details

- **Landing Page**
  - Basic Mode - Validate label(s) against an LGR
  - Review Mode - Review IDN table(s)
  - Advanced LGR Tools
  - Generate ICANN IDN Report
  - Manage IDN review tool
- **Validate label(s) against an LGR**
  - Validate label against RZ-LGR
  - View LGR as XML and HTML
  - Validate a single label or list of label(s)
  - Check label(s) for collision with existing TLD
- **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.)

- Create LGR
  - Create LGR
  - Enter LGR Details
  - Add Code Points
  - Expand Ranges
  - Add References
  - Define Meta Data
  - 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

- Review Impact on Existing Labels by Revising an Existing LGR
  - Select “Diff labels of two LGRs”
  - Enter Details
Agenda Details (cont.)

- Email Notification
- Download Results
- **Review Impact on Existing Labels by Introducing a New LGR**
  - Select “Get collisions in a list of labels”
  - Enter Details
  - Email Notification
  - Download Results
- **Validate Multiple Labels using a single or a merged LGR**
  - Select “Generate disp. annotations”
  - Enter Details
  - Email Notification
  - Download Results
- **View Cross-script Variants of Labels**
  - Select “Cross-script variants”
  - Enter Details
  - Email Notification
  - Download Results
- **Compare LGRs**
  - Compare LGRs
Agenda Details (cont.)

- Select Union, Intersection or Difference
  - With merged LGRs, “Diff” chosen by default

- **Harmonize multiple LGRs**
  - Select “Harmonization”
  - Enter Details
  - Email Notification
  - Download Results

- **Computing variants**
  - Select “Compute Variants”
  - Select relevant LGR
  - Upload list of labels and enter email
  - Download results on email notification
Landing Page

Agenda Item #1
Welcome to the LGR (Label Generation Ruleset) Tools

Select your mode:

- Basic Mode
  - Validate label(s) against an LGR

- Advanced Mode
  - Generate ICANN IDN report
  - Manage IDN review tool

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

© ICANN
Validate label(s) against an LGR – Basic Mode

Agenda Item #2
Validate label(s) against an LGR

Label Validation Tool

Validate label(s) against: RZ-LGR version 4

View LGR as XML | HTML

Upload a list of labels

Check for collision with existing TLDs

Validate

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

⚠️ Note that these files could be cleaned up regularly.
- Download 20210528_105900_annotation lgr-4-arabic-script-29jun20-en.txt.gz
Validate label(s) against an LGR

1. Select required version of RZ-LGR
2. View RZ-LGR version as XML or HTML
3. Two ways to upload label(s)
   3.a. Enter a single label or multiple labels manually
   3.b. Upload a list of labels

Check for collision with existing TLDs

Validate
Validate label(s) against an LGR (Manual Entry of label)

1. Enter label here
2. To add or remove new label
3. Check if label entered collides with existing TLDs
4. "Check for collision" functionality requires email
5. Press "Validate" to get results
Validate label(s) against an LGR (Manual Entry of label)

Validate label(s) against

RZ-LGR version 3

View LGR as XML | HTML

Upload a list of labels

Check for collision with existing TLDs

Validate

INVALID

<table>
<thead>
<tr>
<th>U-label</th>
<th>Disposition</th>
<th>Code point sequence</th>
</tr>
</thead>
<tbody>
<tr>
<td>أبنطنحنرج</td>
<td>invalid</td>
<td>U+0671 (ي) U+06C1 (و) U+0646 (ئ) U+0641 (ف) U+0637 (ط)</td>
</tr>
<tr>
<td>xn-rgbv4adz41a3fyu</td>
<td>invalid</td>
<td>U+0639 (ع) U+062C (ج) U+0641 (ف) U+0686 (ج)</td>
</tr>
</tbody>
</table>
5. To go back to manual entry of labels

1. Upload file encoded in UTF-8 and using Unix line ending

2. Check if file entered collides with existing TLDs

3. “Check for collision” functionality requires email

4. Press “Validate” to get results
Validate label(s) against an LGR (Upload list of label)

Manually input label(s)

Choose File test.txt

File must be encoded in UTF-8 and using UNIX line ending.

Check for collision with existing TLDs

test@test.tld

As the computing may be very long, we will warn by e-mail once the result can be downloaded.

1. Results of list of labels. Refresh page if file does not appear

Your saved results
The following files contains your computation results.

⚠️ Note that these files could be cleaned up regularly.
- Download 20201120_093602_annotation_lgr-3-common-10jul19-en.txt.gz
Advanced LGR Tools

Agenda Item #3
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)
Previously, you edited the following LGR file(s). Click on its title to resume your editing session.
LGRs
- View sample-french
- View union-of-sample-french-and-igr-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_igr-4-arabic-script-29jun20-en.txt.gz

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.

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 igr-1-common-24feb16-en
- Open igr-2-common-26jul17-en
Import or Load LGR

Agenda Item #4
To start by using an existing LGR file in XML format, click on the “Import” button.

Import or Load LGR

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
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

<table>
<thead>
<tr>
<th>Code point</th>
<th>Character Name</th>
<th>Tags</th>
<th>Comments</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>U+0B02 (7) 2 Variant(s)</td>
<td>ORIYA SIGN ANUSVARA</td>
<td>not part of repertoire</td>
<td>See code point</td>
<td></td>
</tr>
<tr>
<td>U+0B03 (6) 2 Variant(s)</td>
<td>ORIYA SIGN VISARGA</td>
<td>not part of repertoire</td>
<td>See code point</td>
<td></td>
</tr>
<tr>
<td>U+0B20 (8) 2 Variant(s)</td>
<td>ORIYA LETTER TTHA</td>
<td>not part of repertoire</td>
<td>See code point</td>
<td></td>
</tr>
<tr>
<td>U+0B9C (6) 2 Variant(s)</td>
<td>TAMIL LETTER JA</td>
<td>not part of repertoire</td>
<td>See code point</td>
<td></td>
</tr>
<tr>
<td>U+0BAE (9) 2 Variant(s)</td>
<td>TAMIL LETTER MA</td>
<td>not part of repertoire</td>
<td>See code point</td>
<td></td>
</tr>
<tr>
<td>U+0BB5 (8) 2 Variant(s)</td>
<td>TAMIL LETTER VA</td>
<td>not part of repertoire</td>
<td>See code point</td>
<td></td>
</tr>
<tr>
<td>U+0BBF (5) 2 Variant(s)</td>
<td>TAMIL VOWEL SIGN I</td>
<td>not part of repertoire</td>
<td>See code point</td>
<td></td>
</tr>
<tr>
<td>U+0BC5 (10) 2 Variant(s)</td>
<td>TAMIL VOWEL SIGN E</td>
<td>not part of repertoire</td>
<td>See code point</td>
<td></td>
</tr>
<tr>
<td>U+0BC7 (11) 2 Variant(s)</td>
<td>TAMIL VOWEL SIGN EE</td>
<td>not part of repertoire</td>
<td>See code point</td>
<td></td>
</tr>
</tbody>
</table>
Import Multiple LGRs

Agenda Item #5
Import or Load LGRs

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

Advanced LGR (Label Generation Ruleset) Tools

This application provides a convenient interface for browsing, creating, loading, and manipulating 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
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 an LGR

Agenda Item #6
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
- Start with a New blank XML file

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

Click on “Add code points” to add code points to the newly created LGR.
Add Code Points

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
3. Click on “Next” button

Either code point value or the character can be added
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)”

<table>
<thead>
<tr>
<th>Code point</th>
<th>Comments</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>U+1780 (ﾘ) 0 Variant(s)</td>
<td>KHMER LETTER KA</td>
<td>See code point</td>
</tr>
<tr>
<td>U+1781 (ﾙ) ... U+1784 (ｵ)</td>
<td>KHMER LETTER KHA ... KHMER LETTER NGO</td>
<td>See code point</td>
</tr>
<tr>
<td>U+1787 (ｶ) 0 Variant(s)</td>
<td>KHMER LETTER CO</td>
<td>See code point</td>
</tr>
<tr>
<td>U+1788 (ｶ) 0 Variant(s)</td>
<td>KHMER LETTER CHO</td>
<td>See code point</td>
</tr>
<tr>
<td>U+1789 (ｷ) 0 Variant(s)</td>
<td>KHMER LETTER NYO</td>
<td>See code point</td>
</tr>
<tr>
<td>U+178A (ｶ) 0 Variant(s)</td>
<td>KHMER LETTER DA</td>
<td>See code point</td>
</tr>
<tr>
<td>U+178B (ｷ) 0 Variant(s)</td>
<td>KHMER LETTER TTHA</td>
<td>See code point</td>
</tr>
<tr>
<td>U+178D (ｷ) 0 Variant(s)</td>
<td>KHMER LETTER TTHO</td>
<td>See code point</td>
</tr>
</tbody>
</table>

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

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

"Existing references" tab shows the added references

<table>
<thead>
<tr>
<th>Reference Id</th>
<th>Description</th>
<th>URL</th>
</tr>
</thead>
</table>

Save

New reference

<table>
<thead>
<tr>
<th>Reference Id</th>
<th>Description</th>
<th>URL</th>
</tr>
</thead>
</table>
Define Meta Data

Click on “Meta data” tab to add meta information about the LGR

<table>
<thead>
<tr>
<th>Code points</th>
<th>References</th>
<th>Meta data</th>
<th>Tags</th>
<th>Rules</th>
</tr>
</thead>
<tbody>
<tr>
<td>Version</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Version comment</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Date</td>
<td>2018-08-08</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Language</td>
<td>und-Khmer</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Scope</td>
<td>example</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Scope type</td>
<td>domain</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Validity start</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Validity end</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Unicode version</td>
<td>10.0.0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Description</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Define Meta Data

Second half of the “Meta data” 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
View Tags

Click on “Tags” tab to view all tag names and associated code points

<table>
<thead>
<tr>
<th>Tag name</th>
<th>Associated code points</th>
</tr>
</thead>
<tbody>
<tr>
<td>vowel</td>
<td>U+17A7 (8)</td>
</tr>
<tr>
<td>consonant</td>
<td>U+1781 (8) U+1782 (6) U+1783 (6) U+1784 (8) U+1787 (2)</td>
</tr>
</tbody>
</table>

To create a new tag: Add it to a code point first to find it back in this list.
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

<table>
<thead>
<tr>
<th>Code point</th>
<th>Character Name</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>U+1780</td>
<td>KHMER LETTER KA</td>
<td>See code point</td>
</tr>
<tr>
<td>U+1781</td>
<td>KHMER LETTER KHA ...</td>
<td>See code point</td>
</tr>
<tr>
<td>U+1785</td>
<td>KHMER LETTER KO</td>
<td>See code point</td>
</tr>
<tr>
<td>U+1786</td>
<td></td>
<td></td>
</tr>
<tr>
<td>U+1788</td>
<td>KHMER LETTER CHO</td>
<td>See code point</td>
</tr>
<tr>
<td>U+1789</td>
<td>KHMER LETTER NYO</td>
<td>See code point</td>
</tr>
<tr>
<td>U+178A</td>
<td>KHMER LETTER DA</td>
<td>See code point</td>
</tr>
<tr>
<td>U+178B</td>
<td>KHMER LETTER TTHA</td>
<td>See code point</td>
</tr>
</tbody>
</table>
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

<table>
<thead>
<tr>
<th>Code point</th>
<th>Tags</th>
<th>Comments</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>U+1780 ( british</td>
<td></td>
<td>consonant</td>
<td></td>
</tr>
<tr>
<td>0 Variant(s)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>U+1781 ( khmer</td>
<td></td>
<td>consonant</td>
<td></td>
</tr>
<tr>
<td>... U+1784 ( ph</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>khmer letter</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>kho</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>U+1787 ( phil</td>
<td></td>
<td>consonant</td>
<td></td>
</tr>
<tr>
<td>0 Variant(s)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>U+1788 ( thai</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0 Variant(s)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>U+1789 ( thai</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0 Variant(s)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>U+178A ( thai</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0 Variant(s)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>U+178B ( thai</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0 Variant(s)</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
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
If the variant definitions are not symmetric, this button will show up. Click “Populate variants” to automatically populate variant mappings.

The screen looks like this after successfully defining code point properties.
Click on “Output” and select “Download” from the given options to download the created LGR.
Validate LGR

Agenda Item #7
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

**Generate stats**

**General stats**
- Number of code points: 113
- Number of ranges: 5
- Largest range: U+076E (Agregar) to U+0771 (Alef) (length: 4)
- Number of sequences: 0

**Variants**
- Total number of variant mappings: 192
- Average number of variants per code point: 3.6
- Largest variant set: 8
- Variants per type:
  - Number of variants for type allocatable: 26
  - Number of variants for type blocked: 166

**Tags**
- Number of code points for tag sc:Arab: 113

**Rules**
- Number of defined rules: 17

Second half of the “Validate LGR” output

Summarized LGR
View LGR as XML and HTML

Agenda Item #8
Click on “Output” and select “View XML” from the given options to get an XML view of the LGR.
View LGR As XML

This XML file does not appear to have any style information associated with it. The document tree is shown below.

```
<lgr xmlns="urn:ietf:params:xml:ns:lgr-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>  <p>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 LAKKHANDAYAO) 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.</p>  <h2>Variants</h2>  <p>According to Section 6 "Variants", in "[Proposal]", this LGR defines no variants.</p>  <h2>Character Classes</h2>  <p>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.</p>  <p>There are 44 characters that are classified as consonants, code points from this subset have been given the tag "cons".</p>  <p>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).</p>  <p>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".</p>  <p>There are 3 diacritical 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].</p>  </l1>  <l1>U+0E47 (MAITAIKHU) and U+0E4C (THAN'THAKHAT) are commonly used in everyday communicating words</l1>  <l1>U+0E4D (NIKHAHIT) is included because of its use to decompose U+0E33 (SARA AM, AM"), which is not otherwise used as a LGR character. Further details may be found in the "[Proposal].</p>
```
Click on “Output” and select “HTML Output” from the given options to view the HTML output of the LGR.
### Meta data in LGR

This document is mechanically formatted from the XML file for the LGR. It provides additional summary data and explanatory text. The XML file remains the sole normative specification of the LGR.

<table>
<thead>
<tr>
<th>LGR Version</th>
<th>2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Date</td>
<td>2017-10-21</td>
</tr>
<tr>
<td>Language(s)</td>
<td>und-Ethi und-Lao ord-thai</td>
</tr>
<tr>
<td>Scope(s)</td>
<td>domain: .</td>
</tr>
<tr>
<td>Unicode Version</td>
<td>6.3.0</td>
</tr>
</tbody>
</table>

### 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 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.

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 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
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>侘</td>
<td>Thai</td>
<td>THAI CHARACTER KO KAI</td>
<td>Thai,und-Thai-cons</td>
<td></td>
<td></td>
<td></td>
<td>[5], [100], [101]</td>
</tr>
<tr>
<td>2</td>
<td>U+0E02</td>
<td>ฅ</td>
<td>Thai</td>
<td>THAI CHARACTER KHO KHAI</td>
<td>Thai,und-Thai-cons</td>
<td></td>
<td></td>
<td></td>
<td>[5], [100], [101]</td>
</tr>
<tr>
<td>3</td>
<td>U+0E03</td>
<td>ฅ</td>
<td>Thai</td>
<td>THAI CHARACTER KHO KHUAT</td>
<td>Thai,und-Thai-cons</td>
<td></td>
<td></td>
<td></td>
<td>[5], [100], [101]</td>
</tr>
<tr>
<td>4</td>
<td>U+0E04</td>
<td>ฅ</td>
<td>Thai</td>
<td>THAI CHARACTER KHO KHWAI</td>
<td>Thai,und-Thai-cons</td>
<td></td>
<td></td>
<td></td>
<td>[5], [100], [101]</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>↔</td>
<td>blocked</td>
<td>[4]</td>
</tr>
<tr>
<td>2</td>
<td>U+1200</td>
<td>U</td>
<td>U+1280</td>
<td>♂</td>
<td>↔</td>
<td>blocked</td>
<td>[4]</td>
</tr>
<tr>
<td>3</td>
<td>U+1210</td>
<td>♂</td>
<td>U+1280</td>
<td>♂</td>
<td>↔</td>
<td>blocked</td>
<td>[4]</td>
</tr>
</tbody>
</table>
### HTML Output - Classes

#### 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>Count</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>14</td>
<td>{U+0EB1 U+0E87 U+0E8A U+0E8D U+0E94 U+0E97 U+0E99 U+0E9A U+0E9F U+0EA1 U+0E9A 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>27</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+0E9B U+0E9C ...}</td>
<td></td>
<td></td>
</tr>
<tr>
<td>und-Laoo-semi-consonant</td>
<td>Tag = und-Laoo-semi-consonant</td>
<td>1</td>
<td>{U+0EBC}</td>
<td></td>
<td></td>
</tr>
<tr>
<td>und-Laoo-tone-mark</td>
<td>Tag = und-Laoo-tone-mark</td>
<td>4</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>7</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>2</td>
<td>{U+0EB8 U+0EB9}</td>
<td></td>
<td></td>
</tr>
<tr>
<td>und-Thai-above-vowel</td>
<td>Tag = und-Thai-av</td>
<td>5</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-vowel</td>
<td>Tag = und-Thai-bv</td>
<td>2</td>
<td>{U+0E38 U+0E39}</td>
<td></td>
<td>Any below vowel</td>
</tr>
<tr>
<td>und-Thai-cons</td>
<td>Tag = und-Thai-cons</td>
<td>44</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>Any consonant or tone</td>
<td></td>
</tr>
<tr>
<td>und-Thai-ct</td>
<td>Tag = und-Thai-ct</td>
<td>48</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>Any consonant or tone</td>
<td></td>
</tr>
<tr>
<td>und-Thai-taa</td>
<td>([und-Thai-consonant:]:[und-Thai-taa:]:[und-Thai-sara-aa:])</td>
<td>49</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>Any consonant, tone or sara-aa</td>
<td></td>
</tr>
<tr>
<td>und-Thai-sara-aa</td>
<td>Tag = und-Thai-sara-aa</td>
<td>1</td>
<td>{U+0E32}</td>
<td>SARA AA</td>
<td></td>
</tr>
<tr>
<td>und-Thai-tone</td>
<td>Tag = und-Thai-tone</td>
<td>4</td>
<td>{U+0E48 U+0E49 U+0E4A U+0E4B}</td>
<td></td>
<td>Any tone mark</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]:[:class property:gc=Mc]:]</td>
<td>True</td>
<td>False</td>
<td>False</td>
<td>False</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></td>
<td>WLE Rule No. 1; semi-consonant must follow a consonant</td>
</tr>
<tr>
<td>und-Laoo-precedes-consonant</td>
<td>← ([[:und-Laoo-consonant:]])</td>
<td>False</td>
<td>True</td>
<td>True</td>
<td></td>
<td>WLE Rule No. 2; vowel-before precedes a main consonant cluster</td>
</tr>
<tr>
<td>und-Laoo-follows-main-consonant</td>
<td>([[:und-Laoo-consonant:]]) ←</td>
<td>False</td>
<td>True</td>
<td>True</td>
<td></td>
<td>WLE Rule No. 3; vowel-above, and vowel-below follow a main consonant C</td>
</tr>
<tr>
<td>und-Laoo-follows-C-tonemark-vabove</td>
<td>([[:und-Laoo-consonant:]]) ←</td>
<td>False</td>
<td>True</td>
<td>True</td>
<td></td>
<td>WLE Rule No. 4; vowel-after follows a main consonant, tone-mark or vowel-above</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>False</td>
<td>Defining consonant cluster for Rule No. 5</td>
</tr>
<tr>
<td>und-Laoo-follows-vbefore-consonant-cluster</td>
<td>([[:und-Laoo-consonant-cluster:]]) ←</td>
<td>False</td>
<td>True</td>
<td>True</td>
<td></td>
<td>WLE Rule No. 5; The sequence 0EOB2 0EOB0 follows a vowel before, and a consonant cluster</td>
</tr>
<tr>
<td>und-Laoo-follows-C-vabove-vbelow</td>
<td>([[:und-Laoo-consonant:]]) ←</td>
<td>False</td>
<td>True</td>
<td>True</td>
<td></td>
<td>WLE Rule No. 6; A tone-mark follows a main consonant, vowel-above or vowel-below</td>
</tr>
<tr>
<td>und-Laoo-follows-Cf</td>
<td>([[:und-Laoo-Cf:]]) ←</td>
<td>False</td>
<td>True</td>
<td>True</td>
<td></td>
<td>WLE Rule No. 7; The sign 0ECC can only occur after final consonants</td>
</tr>
<tr>
<td>und-Laoo-repetition-mark-limit</td>
<td>← ({U+0EC6}{0,2}{end})</td>
<td>False</td>
<td>True</td>
<td>True</td>
<td></td>
<td>WLE Rule No. 8; The sign 0EC6 can only occur 0 to 3 times at the end of the label</td>
</tr>
<tr>
<td>und-Thaip-precedes-consonant</td>
<td>← ([[:und-Thaip-consonant:]])</td>
<td>False</td>
<td>True</td>
<td>True</td>
<td></td>
<td>WLE 7.2: check if current cp is preceding a consonant</td>
</tr>
<tr>
<td>und-Thaip-follows-consonant</td>
<td>([[:und-Thaip-consonant:]) ←</td>
<td>False</td>
<td>True</td>
<td>True</td>
<td></td>
<td>WLE 7.3: check if current cp is following a consonant</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></td>
</tr>
<tr>
<td>2</td>
<td>if at least one variant is in</td>
<td>out-of-repertoire-var</td>
<td>invalid</td>
<td></td>
<td>any variant label with a code point out of repertoire is invalid</td>
</tr>
<tr>
<td>3</td>
<td>if label match</td>
<td>Common-leading-combining-mark</td>
<td>invalid</td>
<td></td>
<td>labels must not commence with a combining mark €</td>
</tr>
<tr>
<td>4</td>
<td>if at least one variant is in</td>
<td>out-of-repertoire-var</td>
<td>invalid</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></td>
</tr>
<tr>
<td>6</td>
<td>if at least one variant is in</td>
<td>out-of-repertoire-var</td>
<td>invalid</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 14,950 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


[2] Corpus Analysis performed by crawling 598 html and 40 PDF files with Tigrigna Contents published online, August 2016

[3] Corpus Analysis performed by crawling 14,850 html Amharic Contents of size 1.8 GB published online, August 2016


[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

Agenda Item #9
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
**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.

<table>
<thead>
<tr>
<th>U-label</th>
<th>Disposition</th>
<th>Code point sequence</th>
</tr>
</thead>
<tbody>
<tr>
<td>p6c1bq3e6e</td>
<td>valid</td>
<td>U+0E00 (c) U+0EAB (m) U+0E94 (c) U+0E81 (m) U+0EB2 (c) U+0E99 (u)</td>
</tr>
</tbody>
</table>

[Action index: 4] <action disp="valid" comment="catch all e"/>

**Variant labels (including original as last)**

1 variant label(s) generated.
By disposition: Counter("valid": 1))

<table>
<thead>
<tr>
<th>U-label</th>
<th>Disposition</th>
<th>Code point sequence</th>
</tr>
</thead>
<tbody>
<tr>
<td>p6c1bq3e6e</td>
<td>valid</td>
<td>U+0E00 (c) U+0EAB (m) U+0E94 (c) U+0E81 (m) U+0EB2 (c) U+0E99 (u)</td>
</tr>
</tbody>
</table>

[Action index: 4] <action disp="valid" comment="catch all e"/>
Validate a Label with a Merged LGR

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.

<table>
<thead>
<tr>
<th>Code point</th>
<th>U-label</th>
<th>Disposition</th>
<th>Code point sequence</th>
</tr>
</thead>
<tbody>
<tr>
<td>U+0067(e)</td>
<td>und-Lao</td>
<td>valid</td>
<td>U+0EC0(e) U+0EAB(m) U+0E94(o) U+0EB1(n) U+0EB2(n) U+0E99(x)</td>
</tr>
<tr>
<td>U+0069(h)</td>
<td>p6c1bc3eb6e</td>
<td>valid</td>
<td></td>
</tr>
</tbody>
</table>

**Valid**

**No collision**

Collision

No collision.

Variant labels (including original as last)

1 variant label(s) generated.
By disposition: Counter("valid": 1).

Variant label:

<table>
<thead>
<tr>
<th>U+0EC0(e) U+0EAB(m) U+0E94(o) U+0EB1(n) U+0EB2(n) U+0E99(x)</th>
<th>p6c1bc3eb6e</th>
</tr>
</thead>
</table>
Review Impact on Existing Labels by Revising an Existing LGR

Agenda Item #10
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. Enter the email address to receive the result notification
5. Check collisions if you want to check label collisions as well
6. Check “Output rules” if you want to check output rules for each label
7. Click on “Get diff” button
Hi,
The processing of diff from labels provided in the file 'test-labels.txt' between LGR 'proposed-arabic-lgr-18092016-en' and LGR 'proposed-arabic-lgr-18092017-en' has been successfully completed. You should now be able to download it from your home screen under the name: '20180823_032644_diff_proposed-arabic-lgr-18092016-en_proposed-arabic-lgr-18092017-en.txt.gz'. Please refresh the home page if you don't see the link.
Best regards
1. 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

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

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

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
# Results

# Labels not in LGR 1 #

# Labels not in LGR 2 #

Label：عملية
Label：نادي

# LGR comparison #
Label：عملية not in LGR proposed-arabic-lgr-18092017-en
Label：نادي not in LGR proposed-arabic-lgr-18092017-en

## Comparison on label 'کلکتت' [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 'نادي' [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

Agenda Item #11
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. Enter the email address to receive the result notification

4. Check “Full Dump” to get summary of each operation done on the labels

5. Check “Output rules” to get rules that the label has gone through

6. Click on “Get collisions”
Hi,
The processing of collisions from labels provided in the file 'test-labels.txt' in LGR 'proposed-arabic-lgr-18092017-en' has been successfully completed. You should now be able to download it from your home screen under the name: '20180823_031221_collisions_proposed-arabic-lgr-18092017-en.txt.gz'. Please refresh the home page if you don't see the link.
Best regards

Email that gives information about the 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

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

- Download 20210528_102129_collisions_lgr-4-arabic-script-29Jun20-en.txt.gz

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

# Labels not in LGR #

<table>
<thead>
<tr>
<th>Label</th>
<th>Code points</th>
<th>Category</th>
</tr>
</thead>
<tbody>
<tr>
<td>[U+06AA U+0644 U+0643 U+062A U+06C1]</td>
<td>Primary</td>
<td></td>
</tr>
<tr>
<td>[U+06A9 U+0644 U+0643 U+062A U+06C1]</td>
<td>Primary</td>
<td></td>
</tr>
</tbody>
</table>

# Collisions #

## Collision ##

```
Label:
Code points: [U+06AA U+0644 U+0643 U+062A U+06C1] | [U+06A9 U+0644 U+0643 U+062A U+06C1]
Category: Primary
```

## Collision ##

```
Label:
Code points: [U+06AA U+0644 U+0643 U+062A U+06C1] | [U+06AA U+0644 U+0643 U+062A U+0629]
Category: Primary | Variant
```

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

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

### Details for label 'ستكتم' [U+06A9 U+0644 U+0643 U+062A U+06C1] ###

```
Variant 'ستكتم' [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 LGR(s)

Agenda Item #12
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. Enter the email address to receive the result notification

4. 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. Enter the email address to receive the result notification

5. Click on “Annotate” button
Hi,
The processing of annotation from labels provided in the file 'lao-test-labels.txt' in LGR 'proposal-lao-lgr-31jan17-en' has been successfully completed.
You should now be able to download it from your home screen under the name: '20180823_033613_annotation_proposal-lao-lgr-31jan17-en.txt.gz'.
Please refresh the home page if you don't see the link.
Best regards
Download 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
- 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,

Click on the download link to get the “Generate disp. annotations” 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.
View Cross-script Variants of Labels

Agenda Item #13
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. Enter the email to receive the result notification

5. Click on “Launch” button
Hi,
The processing of cross-script variants from labels provided in the file ‘test-labels.txt’ in LGR ‘proposed-armenian-lgr-05nov15-en’ has been successfully completed. You should now be able to download it from your home screen under the name: ‘20180823_073155_cross_script_variants_proposed-armenian-lgr-05nov15-en.txt.gz’. Please refresh the home page if you don’t see the link.
Best regards

Email that gives information about the results
Download 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

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.

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 (ωω) has cross-script variants:
- Cross-variant U+0448 U+0575 U+0580 (ωω), disposition invalid:
  + U+0448 (ω): Cyril
Input label U+0581 U+0561 U+0579 U+0586 (γωγ) has cross-script variants:
- Cross-variant U+0581 U+0448 U+0579 U+0586 (γωγ), disposition invalid:
  + U+0448 (ω): Cyril
- Cross-variant U+0067 U+0561 U+0579 U+0586 (γωγ), disposition invalid:
  + U+0067 (γ): Latin
- Cross-variant U+0067 U+0448 U+0579 U+0586 (γωγ), disposition invalid:
  + U+0448 (ω): Cyril
  + U+0067 (γ): Latin
Input label U+0566 U+0578 U+057D (𝑞𝑛𝑢) has cross-script variants:
- Cross-variant U+0566 U+0578 U+0075 (𝑞𝑛𝑢), disposition invalid:
  + U+0075 (𝑢): Latin
- Cross-variant U+0566 U+006E U+057D (𝑞𝑛𝑢), disposition invalid:
  + U+006E (𝑛): Latin
- Cross-variant U+0566 U+006E U+0075 (𝑞𝑛𝑢), disposition invalid:
  + U+0075 (𝑢): Latin
  + U+006E (𝑛): Latin
- Cross-variant U+0071 U+0578 U+057D (𝑞𝑛𝑢), disposition invalid:
  + U+0071 (𝑞): Latin
- Cross-variant U+0071 U+0578 U+0075 (𝑞𝑛𝑢), disposition invalid:
  + U+0071 (𝑞): Latin
  + U+0075 (𝑢): Latin
- Cross-variant U+0071 U+006E U+057D (𝑞𝑛𝑢), disposition invalid:
  + U+0071 (𝑞): Latin
  + U+006E (𝑛): Latin
- Cross-variant U+0071 U+006E U+0075 (𝑞𝑛𝑢), disposition invalid:
  + U+0071 (𝑞): Latin
  + U+0075 (𝑢): Latin
  + U+006E (𝑛): Latin
Compare LGRs

Agenda Item #14
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

<table>
<thead>
<tr>
<th>First LGR</th>
<th>merged-lgr-1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Second LGR</td>
<td>merged-lgr-2</td>
</tr>
</tbody>
</table>

Note that 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

** Result of diff of proposed-arabic-lgr-18092016-en with proposed-arabic-lgr-18092017-en **

** Compare Metadata **
Compare Description:
Second LGR has no description

Languages values differ:
Values only in first LGR: ['und-Arab'].
Values only in second LGR: [].

** Compare repertoire **
Repertoire values differ:
Values only in first LGR: U+062F U+063A U+0635 U+0636 U+0637 U+0630 U+0631 U+062C U+0632 U+062D U+0638 U+0633 U+062E U+0639 U+0634.
Values only in second LGR: .

** Compare common code points in repertoire **
113 code points are identical

** Compare WLE **

Differences of two LGRs

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

Agenda Item #15
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

The Harmonized LGRs will also be listed on the homepage.
Computing Variants

Agenda Item # 16
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. Email id for notification

4. Press this to compute variants
Press here to go to homepage. The results will be present on the homepage when available.

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

- 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_092125_labels_variants_lgr-4-devanagari-script-29jun20-en.csv.gz

Download results of compute variants
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