Frogans Technology OP3FT
FACR 1.0 December 4, 2014
Adopted
ISBN 978-2-37313-001-0

Frogans Address Composition Rules - 1.0

Abstract

This document sets forth the composition rules applicable to Frogans addresses. These rules focus on security. They manage language-related issues by introducing the concepts of linguistic categories and convergence forms. The composition rules apply to Frogans addresses that are compliant with the pattern defined in the International Frogans Address Pattern (IFAP) specification. These rules are enforced by the FCR Operator at the time a Frogans address or a Frogans network is added to the FCR.

Status

This document is an official technical specification of the Frogans technology.

This technical specification was adopted by the OP3FT on December 4, 2014.

Comments on this document are welcome and may be made on the Frogans technology mailing lists, accessible at the following permanent URL: https://lists.frogans.org/.

Location

This document is accessible at the following permanent URL: https://www.frogans.org/en/resources/facr/access.html.

Copyright Statement

Table of Contents

1. Introduction

1.1. Background

Started in 1999, the Frogans project aims to introduce a new software layer on the Internet alongside other existing layers such as E-mail or the Web. The goal of this new software layer, called the Frogans layer, is to enable the publishing of Frogans sites.

The Frogans technology developed for the Frogans project is the foundation of the Frogans layer. It involves using Frogans addresses and Frogans networks that are registered in a central database, called the Frogans Core Registry (FCR).

A separate technical specification, International Frogans Address Pattern (IFAP) [IFAP], describes the pattern applicable to Frogans addresses, including their structure based on network names and site names. IFAP describes Frogans addresses from a technical standpoint and is designed to be language-independent.

The IFAP specification defines the character set of the Unicode Standard [Unicode] as the character set used to represent Frogans address strings.

Frogans addresses are designed to support international characters used in a wide range of languages and writing systems. While this gives Frogans address and Frogans network holders more freedom in choosing network names and site names, it also raises potential security issues for end users.

The most important issue relates to spoofing, whereby a malicious person would attempt to mislead end users by choosing a Frogans address that could be confused with a legitimate registered Frogans address.

In order to mitigate this type of security issue related to international identifiers, including Internationalized Domain Names (IDNs), extensive work has already been carried out by organizations such as the Unicode Consortium, the World Wide Web Consortium (W3C), the IETF, ICANN, and various domain name registry operators.

This work has shown that end-user confusion between international identifiers can occur at several levels, such as:

* Confusion between characters, or sequences of characters, belonging to a given writing system. For example, in the Latin writing system, the following characters have a similar visual appearance: U+0049 LATIN CAPITAL LETTER I, U+006C LATIN SMALL

LETTER L, and U+0031 DIGIT ONE.

- Confusion between characters, or sequences of characters, belonging to different writing systems. For example, the following characters have a similar visual appearance: U+0430 CYRILLIC SMALL LETTER A in the Cyrillic writing system and U+0061 LATIN SMALL LETTER A in the Latin writing system.

- Confusion between characters, or sequences of characters, belonging to a language that has two different writing systems. For example, in the Chinese language, the following characters are considered as possibly confusing for end users: U+5B81 corresponding to "calm, peaceful, serene; healthy" in Simplified Chinese, and U+5BE7 corresponding to "repose, serenity, peace; peaceful" in Traditional Chinese. Characters of this kind are often called "variants".

In the preceding examples, as well as in the remainder of this specification, the Unicode code points corresponding to characters are represented using the "U+code" format, where "code" is a series of four to six uppercase hexadecimal digits representing the numerical value of the code point.

Compared to Internationalized Domain Names (IDNs), Frogans addresses present an additional risk of confusion for end users since, unlike domain names, the Frogans address pattern supports the use of uppercase characters.

1.2. Purpose

The purpose of this document is to set forth the rules for the composition of Frogans addresses.

This FACR specification that deals with security issues, notably concerning support for multiple languages, is called for in the OP3FT Bylaws [BYLAWS].

This FACR specification is complementary to the IFAP specification [IFAP]. The two-part model for specifying Frogans addresses and its benefits are presented in the IFAP specification (see IFAP, section 1.4 Stability and Security).

The rules in FACR are enforced by the FCR Operator at the time a Frogans address or a Frogans network is added to the FCR. They are applied to network names and site names that are already compliant with the IFAP specification.

Since the FCR is a multilingual registry, the rules in FACR must obviously take into account, and combine, the outcome of various works, contributed previously to the community, concerning international identifiers. This work, which implicates linguists and other experts from around the world, is a tremendous task and is still in progress for most languages and writing systems at the time this FACR specification is being completed. For instance, as concerns Internationalized Domain Names (IDNs), ICANN issued a call [GPCALL] in July 2013 for the formation of community panels, referred to as Generation Panels, to establish rules for each language or writing system regarding the characters that are acceptable for top-level domains (TLDs), and to manage any variant labels. Some community panels have already started their work.

New versions of this FACR specification will be prepared as needed in order to take into account work that will be contributed to the community in the future, concerning international identifiers.

In order to allow Frogans address composition rules to evolve quickly and easily over time, as required by the two-part model, this specification must define a flexible and modular architecture for FACR.

1.3. Intended Audience

This document is intended for those involved in the Frogans address registration process, such as Frogans address holders, FCR account administrators, and the Operator of the Frogans Core Registry (FCR).

For example, Frogans address holders can use this document to understand the composition rules applicable to their Frogans addresses.

This document is also intended for developers wishing to implement software related to Frogans address registration, and in general for anyone interested in the security model underpinning the addressing system used for Frogans sites.

To comprehend the choices made in this specification, it is necessary to understand the context in which these choices are made. This is not an easy task, since the multiple standards and specifications underlying the Frogans address composition rules require time and effort to assimilate and use correctly.

Therefore, in order to make this specification accessible to the widest possible audience, it was decided to provide, when required, relevant background information before describing the choices made. As a result, this specification often alternates background information and rules applicable to Frogans addresses. The background information may include a detailed reference to the underlying standard or specification.

In addition, the appendices provide assistance in implementing certain parts of this specification. They contain lookup tables with pre-processed lists of code points (Appendix A), pseudocode syntax (Appendix B), and a series of verification and generation processes (Appendix C). The goal is to avoid the need for developers to access and analyze the data and the algorithms defined in the multiple standards and specifications involved in Frogans address composition rules.

1.4. Supporting Policies

The security model underpinning the registration of Frogans addresses and Frogans networks in the FCR cannot be ensured solely through technical means such as the IFAP and FACR specifications. Any purely technical approach could never be entirely successful.

For example, an unscrupulous individual could register Frogans addresses which, although compliant with the existing rules in this FACR specification, would be chosen intentionally to mislead end users. The history of domain name management includes numerous examples demonstrating the creativity of malicious individuals when it comes to spoofing. An individual could also abusively register a Frogans address that could legitimately be registered by another company or organization.

Therefore policies are required in order to anticipate and react to inappropriate behavior concerning Frogans address registration that cannot be prevented on a purely technical level. To that end, in addition to technical specifications, the OP3FT has defined two enforceable policies that play an important role in the overall security environment of the Frogans addressing system:

* Frogans Technology User Policy [FTUP]: this policy notably defines the rights and obligations of Frogans address and Frogans network holders. In particular, it prohibits malevolent actions that cannot be restricted from a purely technical standpoint, either temporarily or permanently.

  To facilitate the application of this policy as concerns Frogans address registration, this policy requires that the FCR Operator provide the following two services:

  - FCR Whois database query service: the FCR Whois database can be queried by anyone in order to verify the identity of the holder of any Frogans address or Frogans network and to retrieve contact information.

- FCR public data download service: the FCR public data includes all registered Frogans addresses and Frogans networks. It can be downloaded by anyone. Organizations such as trademark monitoring service providers can use this data in order to provide monitoring services on Frogans address registrations.

* Uniform Dispute Resolution Policy for Frogans Addresses [UDRP-F]: this policy is designed to protect trademark holders from abusive registration of Frogans addresses and Frogans networks in the FCR. This policy, along with its Rules, sets out the legal framework for resolving disputes over the registration and use of a network name or site name.

End-user awareness is another important factor in the overall security environment of the Frogans addressing system. For example, the Frogans Player software, provided by the OP3FT, ensures that the linguistic category, the network name, and site name of a Frogans address can always be easily displayed, thereby enabling end users to clearly identify the Frogans sites they are browsing.

1.5. Compliance

The rules applicable to Frogans addresses in this specification are defined in succession. The definition of each rule assumes compliance with all preceding rules.

A conforming implementation of this specification is an implementation which is compliant with all descriptions appearing in this document, except for:

* descriptions in paragraphs that do not directly concern the Frogans technology, but provide background information intended to help understand the context and the reasons for choices made

* descriptions found in sections that are indicated as not normative, such as the appendices which provide assistance in implementing certain parts of this specification

* descriptions in the form of examples that illustrate certain aspects of the specification

Hence, unlike in specifications elaborated by several other organizations, requirement levels in this specification are not indicated using key words such as "must", "must not", "should", and "should not" defined in RFC 2119 [RFC2119]. This applies to all specifications elaborated by the OP3FT.

Normative and informative references appear between square brackets [] in this document. Their details are included in the References section.

2. Terminology

This section defines key terms used in this specification, listed so as to facilitate their comprehension when read in the order presented.

OP3FT

A non-profit organization whose purpose is to hold, promote, protect, and ensure the progress of the Frogans technology in the form of an open standard for the Internet, available to all, free of charge.

Frogans technology

A secure technology used to implement a new software layer on the Internet, alongside other existing software layers such as E-mail or the Web. The Frogans technology makes it possible to publish Frogans sites.

Frogans site

A set of Frogans pages, called "slides", hyperlinked to each other, available online on the Internet or in an intranet, at a Frogans address. A Frogans site can be published by any individual or organization, from anywhere in the world, in any language.

Frogans address

A string of characters serving as the identifier of a Frogans site. Frogans addresses include two parts, separated by the asterisk character: the network name and the site name. Frogans addresses may contain international characters and may include uppercase, lowercase, and accented characters. Frogans addresses may be written from left to right or from right to left. For example, in the left-to-right writing direction, the pattern of a Frogans address is "network-name*site-name".

Eligible character

A character that can be used in a Frogans address. Eligible characters are defined in the IFAP specification.

Separator character

The asterisk character. It is used to separate the network name and the site name in a Frogans address.

Network name

The string of characters in a Frogans address that precedes the separator character when writing the Frogans address.

Site name

The string of characters in a Frogans address that follows the separator character when writing the Frogans address.

Connector character

A character that can be used to connect different words included in a network name or a site name. Connector characters are defined in the IFAP specification.

Reference form

Form of a network name, a site name, or a Frogans address generated to evaluate its length and to check whether two network names, site names, or Frogans addresses are identical. This form is not intended for display to end users. The generation of this form is defined in the IFAP specification.

Preferred form

Form of a network name, a site name, or a Frogans address as registered in the Frogans Core Registry by its holder. Frogans Player uses this form to display Frogans addresses to end users.

Frogans network

A group of Frogans addresses that have an identical network name.

Linguistic category

A group of languages using the same writing system, or a language using one or more writing systems. The network name of a Frogans network is associated with a linguistic category. The site name in a Frogans address is associated with the same linguistic category as the network name. Each linguistic category has employable characters and arrangement rules.

Available linguistic category

A linguistic category defined in this specification.

Language

A means used by a group of people to communicate. A language comprises words and methods of combining them.

Writing system

A system used to write a language. A writing system includes graphemes that can represent, for instance, words, syllables, or alphabetic letters. Certain writing systems can be used to write several languages.

Employable character

A character, defined in the context of a linguistic category, that can be used in a network name or a site name. The employable characters of a linguistic category are the same for the network name and for the site name.

Arrangement rule

A rule, defined in the context of a linguistic category, that relates to the arrangement of employable characters in a network name or a site name. The arrangement rules of a linguistic category can be different for the network name and for the site name.

Valid network name

A network name that is valid in the context of a linguistic category, as regards employable characters and arrangement rules.

Valid site name

A site name that is valid in the context of the linguistic category of the network name with which it is used, as regards employable characters and arrangement rules.

Overlapping linguistic categories

Linguistic categories that have valid network names in common.

Convergence form

Form of a valid network name or a valid site name used to check whether two valid network names or two valid site names are excessively similar, or "convergent". This form is not intended for display to end users. There are two kinds of convergence forms: Intra-LC convergence forms and Inter-LC convergence forms.

Intra-LC convergence form

A convergence form, defined in the context of a linguistic category, that is used to check whether two valid network names or two valid site names associated with that linguistic category are convergent. There can be more than one type of Intra-LC convergence form defined in the context of a linguistic category. Intra-LC convergence forms of each type are generated using preferred forms.

Inter-LC convergence form

A convergence form used to check whether two valid network names associated with different linguistic categories are convergent. Inter-LC convergence forms do not apply to site names. There is only one type of Inter-LC convergence form. Inter-LC convergence forms are generated using preferred forms.

Frogans Core Registry, FCR

The database which contains all registered Frogans addresses and Frogans networks. The database belongs to the OP3FT.

FCR Operator

The entity responsible for the technical and commercial operation of the FCR, under a delegation agreement with the OP3FT.

Frogans Player

Free-of-charge software used to browse Frogans sites. Frogans Player is to be made available on a wide range of fixed and mobile devices. It is developed and distributed by the OP3FT.

3. The Need for New Concepts

The rules developed by various organizations to mitigate security issues related to international identifiers are designed to resolve problems that are different in scope.

For example, the Unicode Technical Standard #39 [UTS39] defines methods for determining whether international identifiers are confusable, including sophisticated algorithms involving "Unicode scripts" as well as mappings to manage visually confusable characters. By contrast, CNNIC, the registry operator of the .cn ccTLD, defines methods for determining whether a Chinese domain name (CDN) can be registered, including a list of authorized characters as well as mappings to manage variants in the Chinese language [IDN-CN].

As a result of the difference in scope, these rules work at different levels and hence are based on different rule-integration models, which makes them difficult to combine. In order to be able to integrate these various rules in this FACR specification while keeping the specification easy to upgrade, a specific rule-integration model is needed.

To provide the foundation for this new rule-integration model, two new concepts are introduced:

* linguistic categories, whose purpose is to clarify the language or writing system of each Frogans address

* convergence forms, whose purpose is to detect excessive similarities between Frogans addresses

The introduction of these concepts in the FCR allows Frogans address holders to name their Frogans sites precisely, and allows end users to benefit from secure and easy-to-use addresses.

3.1. Linguistic Categories

A linguistic category can correspond either to:

* a group of languages using the same writing system, or to

* a language using one or more writing systems

Each linguistic category is identified using a unique label characterizing the category. The label is a string of ASCII characters [ASCII] starting with the three characters 'L' (0x4C), 'C' (0x43) and '-' (0x2D), followed by between 3 and 16 characters from 'A' to 'Z' (0x41-0x5A) or from 'a' to 'z' (0x61-0x7A).

The network name of a Frogans network is associated in the FCR with a linguistic category. This association is created at the time the Frogans network is added to the FCR and cannot henceforth be changed.

The site name in a Frogans address is associated with the same linguistic category as the network name in that Frogans address.

As a result, each Frogans address in the FCR is associated with a single linguistic category.

This association between a Frogans network or a Frogans address and a linguistic category can help in the application of the supporting policies presented in Section 1.4. Typically, in the event of inappropriate behavior concerning the registration of a Frogans network or of a Frogans address, the intentions of the holder can be clarified by the choice of the associated linguistic category, which is indicated in the FCR Whois database.

Each linguistic category has:

* employable characters: the characters that can be used in the network name or the site name of a Frogans address, and

* arrangement rules: the rules that govern how the employable characters can be arranged relative to each other in the network name or the site name of a Frogans address

Both the employable characters and the arrangement rules of a linguistic category are necessary to determine the validity of the network name or the site name of a Frogans address associated with that linguistic category.

In order to prevent combinations of languages and writing systems that could lead to confusion in Frogans addresses, care must be taken when defining the list of linguistic categories in conjunction with the employable characters and arrangement rules of each linguistic category.

The linguistic categories are defined in line with the following principles:

* The number of linguistic categories is kept to a minimum: languages that share the same properties are grouped together in the same linguistic category.

* There is no hierarchy between linguistic categories: no linguistic category is a sub-category of another linguistic category.

* A linguistic category is independent of other linguistic categories: the rules concerning both the employable characters and the arrangement rules of a linguistic category can evolve without impacting the rules of other linguistic categories.

* The linguistic category of a Frogans address should be clearly distinguishable by humans or, failing that, by systems, and the possibility that a Frogans address could be associated with more than one linguistic category is kept to a minimum.

As a result of these principles, the territorial variations of a language are grouped in the same linguistic category.

The following sources are used for defining the list of linguistic categories:

* the list of Unicode scripts defined by the Unicode Standard Annex #24 [UAX24] which are used to represent textual information in writing systems

* the list of script names defined in ISO 15924 [ISO15924]

* the lists of languages and language groups defined in ISO 639 [ISO639]

3.2. Convergence Forms

Rules developed by organizations to mitigate security issues related to international identifiers can be used to produce, in a given context, the set of all the identifiers that are excessively similar to a given identifier. This set can be referred to as a "bundle".

These rules can also be used to implement, in a given context, a transform where a given identifier and all its excessively similar identifiers are transformed into the same form.

In a registry of international identifiers, either of these two approaches can be used to enable detection of excessively similar identifiers before a new identifier can be added to the registry. In the first approach, the method of detection requires that for each identifier already registered, the set of all its excessively similar identifiers be stored. In the second approach, the method requires that only the transformed form of the identifier be stored.

In this FACR specification, a single approach applying to all Frogans addresses has to be chosen to make the specification simple and easy to upgrade. Since a set of excessively similar Frogans addresses could potentially contain thousands of Frogans addresses, the second approach, which is advantageous in terms of space provisioning and scalability in the FCR, is chosen. The transformed form of the second approach is called a convergence form.

Convergence forms are used to check whether network names or site names are excessively similar, or "convergent". They are not intended for display to end users.

A convergence form can be either an Intra-LC convergence form, defined in the context of a linguistic category, or an Inter-LC convergence form:

1. Intra-LC convergence forms apply to network names and site names that are associated with a linguistic category.

   Intra-LC convergence forms are used to check whether two network names or two site names associated with the same linguistic category are convergent.

   One or more types of Intra-LC convergence form can be defined in the context of a linguistic category.

2. Inter-LC convergence forms apply to network names, regardless of the linguistic category with which they are associated. They do not apply to site names.

   Inter-LC convergence forms are used to check whether two network names associated with different linguistic categories are convergent.

   Only one type of Inter-LC convergence form is defined.

As a result, for each network name associated with a linguistic category and registered in the FCR, one or more Intra-LC convergence forms and one Inter-LC convergence form are stored. For each site name associated with a linguistic category and registered in the FCR, one or more Intra-LC convergence forms are stored.

Each type of Intra-LC convergence form is defined using the rules developed by one or more organizations to mitigate security issues related to the languages and writing systems corresponding to the linguistic category.

The following sources are used for defining the different types of convergence forms:

- The mechanisms for detecting visually confusable strings, defined in the Unicode Technical Standard #39 [UTS39] (see the Unicode Technical Standard #39, section 4 Confusable Detection).

  This source is used for defining Intra-LC convergence form types and the Inter-LC convergence form type.

- The mechanisms to enforce language-based character variant preferences, defined in RFC 3743 [RFC3743] and used with IDN tables included in the Repository of Internationalized Domain Name (IDN) Practices maintained by the Internet Assigned Numbers Authority (IANA) [IANA-Repository].

  This source is used for defining Intra-LC convergence form types.

4. Rules for Each Linguistic Category

This section describes the rules used to define the employable characters and the arrangement rules of a linguistic category.

These rules are applied to network names and site names that are already compliant with version 1.1 of the International Frogans Address Pattern (IFAP) specification [IFAP], which is the latest available version at the time this FACR specification is being completed. The IFAP specification sets forth rules concerning, for example, string formation, eligible characters, directionality, connector characters, and the length of network names and site names.

This FACR specification uses various mechanisms defined in the Unicode Standard [Unicode], including the Unicode Standard Annexes, as well as in Unicode Technical Standards. Since version 1.1 of the IFAP specification uses version 7.0.0 of the Unicode Standard, this version of FACR also uses version 7.0.0 of the Unicode Standard.

4.1. Employable Characters

The employable characters of a linguistic category are those characters that can be used in a network name or a site name associated with that linguistic category.

The employable characters of a linguistic category are the same for a network name and for a site name associated with that linguistic category.

It is important to note that the international characters used in the contents of a Frogans site are not limited to the employable characters of the linguistic category of the Frogans address identifying that Frogans site.

In order to define the employable characters of a linguistic category, it is necessary to select a source of characters, called the primary source, that meets the following requirements:

1. If the linguistic category corresponds to a group of languages using the same writing system, then the primary source contains all the characters commonly used in these languages.

2. If the linguistic category corresponds to a language using one or more writing systems, then the primary source contains all the characters commonly used in this writing system or these writing systems for this language.

3. The primary source only contains the characters commonly used in the languages or writing systems corresponding to the linguistic category.

4. If the writing system or systems corresponding to the linguistic category include characters with different case forms, then the primary source contains the lowercase form of these characters. The primary source is not required to contain the uppercase and titlecase forms of these characters, since these forms are incorporated in the methods provided in this section for checking whether a code point is accepted as a potential employable character.

In addition to the preceding requirements:

5. The primary source has been created and is maintained by recognized experts in the languages and the writing systems corresponding to the linguistic category.

6. The primary source was contributed by either a government organization or a work group managed under the auspices of a worldwide organization.

7. The primary source is widely adopted as a source for acceptable characters, and has been thoroughly tested.

8. The primary source is usable by all, free of charge, in a perpetual manner and without restriction.

On the basis of these requirements, two types of primary sources are selected for determining the employable characters of linguistic categories:

* IDN tables included in the Repository of Internationalized Domain Name (IDN) Practices maintained by the Internet Assigned Numbers Authority (IANA) [IANA-Repository]

  IDN tables represent permitted characters allowed for IDN registrations in certain Top-Level Domain registries, including country-code Top-Level Domain (ccTLD) registries.

* Data included in the Unicode Common Locale Data Repository (CLDR) maintained by the Unicode Consortium [CLDR]

  CLDR is used by many organizations worldwide. It provides a standard repository of locale data in order to support the world's languages.

The type of the primary source used for determining the employable characters of a linguistic category is selected using the following method:

A. If there is an IDN table that meets all the preceding requirements in this section, then the primary source is this IDN table.

B. Otherwise, the primary source is data included in CLDR.

This method is applied irrespective of whether the linguistic category corresponds to a group of languages using the same writing system, or corresponds to a language using one or more writing systems.

The following two sections describe methods for checking whether a code point is accepted as a potential employable character when using either an IDN table or CLDR as the primary source for determining the employable characters of a linguistic category.

For assistance in implementing a function to verify compliance regarding employable characters, see Appendix C.1.

4.1.1. Using an IDN table as the primary source

This section describes the method for checking whether a code point is accepted as a potential employable character when using an IDN table as the primary source for determining the employable characters of a linguistic category. It is assumed in this section that the code point corresponds to an eligible character.

The method takes the following values as input:

* CP: the code point

* IDNT: the IDN table

The method uses the following terms:

* Simple_Uppercase_Mapping, Simple_Titlecase_Mapping: these terms refer to properties defined in the Unicode Standard Annex #44 [UAX44] (see the Unicode Standard Annex #44, section 5.3 Property Definitions).

The method consists of performing the following tests in succession until it has been determined whether or not CP is accepted as a potential employable character:

A. If CP corresponds to a permitted character in IDNT,

then CP is accepted.

B. If there is a permitted character in IDNT for which the value of either the Simple_Uppercase_Mapping property or the Simple_Titlecase_Mapping property equals CP,

then CP is accepted.

C. Otherwise, CP is not accepted.

In the method, the location of the permitted characters in IDNT depends on the format of that IDN table. For example, in IDN tables based on the format defined in RFC 3743 [RFC3743], the permitted characters, which are referred to as "Valid Code Points", are located in the first column of the IDN table.

4.1.2. Using CLDR as the primary source

This section describes the method for checking whether a code point is accepted as a potential employable character when using CLDR as the primary source for determining the employable characters of a linguistic category. The method is used with a script subtag and an option for including auxiliary exemplar sets. It is assumed in this section that the code point corresponds to an eligible character.

The method takes the following values as input:

* CP: the code point

* SST: the script subtag

* AUX: the option for including auxiliary exemplar sets

The method uses the following terms:

* Simple_Uppercase_Mapping, Uppercase_Mapping, Simple_Titlecase_Mapping, Titlecase_Mapping: these terms refer to properties defined in the Unicode Standard Annex #44 [UAX44] (see the Unicode Standard Annex #44, section 5.3 Property Definitions).

The method consists of performing the following tests in succession until it has been determined whether or not CP is accepted as a potential employable character:

A. If CP corresponds to a character in the main exemplar set of any of the language identifiers associated with SST,

   then CP is accepted.

B. If CP corresponds to a character in the punctuation exemplar set of any of the language identifiers associated with SST,

   then CP is accepted.

C. If CP corresponds to a character in the decimal digit set of any of the language identifiers associated with SST,

   then CP is accepted.

D. If there is a character in any of the preceding sets for which the value of either the Simple_Uppercase_Mapping property, the Uppercase_Mapping property, the Simple_Titlecase_Mapping property, or the Titlecase_Mapping property equals CP,

   then CP is accepted.

E. If AUX is enabled, then three cases can arise:

   1. If CP corresponds to a character in the auxiliary exemplar set of any of the language identifiers associated with SST,

      then CP is accepted.

   2. Otherwise, if there is a character in any of the preceding auxiliary exemplar sets for which the value of either the Simple_Uppercase_Mapping property, the Uppercase_Mapping property, the Simple_Titlecase_Mapping property, or the Titlecase_Mapping property equals CP,

      then CP is accepted.

   3. Otherwise, CP is not accepted.

F. If AUX is disabled, then CP is not accepted.

In the method, the following techniques are used:

* The list of language identifiers associated with SST is created as follows, using the <language> elements contained in the <languageData> element of the CLDR XML data file supplementalData.xml [CLDR].

For each <language> element for which the value of the "scripts" attribute contains SST and the value of the "alt" attribute is not equal to 'secondary':

1. If the value of the "scripts" attribute contains SST only:

    i. If the "territories" attribute is included in the <language> element, then for each region subtag within the value of the "territories" attribute, an item consisting of the concatenation of the value of the "type" attribute and '_' and the region subtag is added to the list.

    ii. Otherwise, if the "territories" attribute is not included in the <language> element, then an item consisting of the value of the "type" attribute is added to the list.

2. Otherwise, if the value of the "scripts" attribute contains SST amongst other script subtags:

    i. If the "territories" attribute is included in the <language> element, then for each region subtag within the value of the "territories" attribute, an item consisting of the concatenation of the value of the "type" attribute and '_' and SST and '_' and the region subtag is added to the list.

    ii. Otherwise, if the "territories" attribute is not included in the <language> element, then an item consisting of the concatenation of the value of the "type" attribute and '_' and SST is added to the list.

All items in the list are language identifiers referred to as Unicode language identifiers in the Unicode Technical Standard #35 [UTS35] (see the Unicode Technical Standard #35, Part 1, Core, 3.1 Unicode Language Identifier).

* The characters in the exemplar sets of a language identifier associated with SST are retrieved from the <exemplarCharacters> elements contained in the <characters> element of the fully-resolved CLDR XML data file [CLDR] associated with that language identifier, using:

  - For the main exemplar set: the <exemplarCharacters> element for which the "type" attribute is not included

- For the punctuation exemplar set: the <exemplarCharacters> element for which the value of the "type" attribute is equal to 'punctuation'

- For the auxiliary exemplar set: the <exemplarCharacters> element for which the value of the "type" attribute is equal to 'auxiliary'

The process used to fully resolve the CLDR XML data file associated with the language identifier is described in the Unicode Technical Standard #35 [UTS35] (see the Unicode Technical Standard #35, Part 1, Core, 4.2.2 Resolved Data File).

The syntax used to convert the content of the <exemplarCharacters> element into the corresponding code points is described in the Unicode Technical Standard #35 [UTS35] (see the Unicode Technical Standard #35, Part 2, General, 3.1 Exemplar Syntax).

* The characters in the decimal digit set of a language identifier associated with SST are retrieved as follows:

  - If, amongst the <numberingSystem> elements contained in the <numberingSystems> element of the CLDR XML data file numberingSystems.xml [CLDR], there is a <numberingSystem> element for which the value of the "type" attribute is equal to 'numeric' and for which the value of the "id" attribute is equal to the content of the <defaultNumberingSystem> element contained in the <numbers> element of the fully-resolved CLDR XML data file associated with that language identifier, then the characters in the decimal digit set are retrieved from the value of the "digits" attribute of that <numberingSystem> element.

  - Otherwise, the decimal digit set is empty.

  The process used to fully resolve the CLDR XML data file associated with the language identifier is described in the Unicode Technical Standard #35 [UTS35] (see the Unicode Technical Standard #35, Part 1, Core, 4.2.2 Resolved Data File).

4.2. Arrangement Rules

The arrangement rules of a linguistic category are rules that relate to the arrangement of employable characters in a network name or a site name associated with that linguistic category.

The arrangement rules of a linguistic category are defined when necessary. They can be different for a network name and for a site name associated with that linguistic category.

The arrangement rules of a linguistic category for a site name have the same outcome irrespective of the preferred form of the network name.

As a result, the site name of a Frogans address continues to comply with all rules in this specification in the event that the preferred form of the network name of the Frogans address is modified.

The following sources are used for defining the arrangement rules of linguistic categories:

* the rules that describe the contexts in which particular characters are permitted, defined in RFC 5892 [RFC5892], which is part of Internationalized Domain Names for Applications [IDNA2008] (see RFC 5892, appendix A Contextual Rules Registry)

* the rules concerning the use of numerals, defined in RFC 5564 [RFC5564] (see RFC 5564, section 2.3.1 Numerals)

* the rules concerning the use of different decimal number systems, defined in the Unicode Technical Standard #39 [UTS39] (see the Unicode Technical Standard #39, section 5.3 Mixed-Number Detection)

* the rules concerning the use of characters, defined in the policy documents of IDN tables included in the Repository of Internationalized Domain Name (IDN) Practices maintained by the Internet Assigned Numbers Authority (IANA) [IANA-Repository]

For assistance in implementing a function to verify compliance regarding arrangement rules, see Appendix C.2.

5. Valid Network Names and Site Names

The validity of a network name or a site name is determined within the context of a linguistic category.

A network name associated with a linguistic category is valid if the network name complies with all of the following:

* the IFAP specification [IFAP]

* the rules that apply to network names concerning the employable characters of that linguistic category

* the rules that apply to network names concerning the arrangement rules of that linguistic category

A site name, used with a valid network name that is associated with a linguistic category, is valid if the site name complies with all of the following:

* the IFAP specification

* the rules that apply to site names concerning the employable characters of that linguistic category

* the rules that apply to site names concerning the arrangement rules of that linguistic category

6. Overlapping Linguistic Categories

A linguistic category overlaps with another linguistic category if these two linguistic categories have valid network names in common. These linguistic categories are said to be overlapping.

The notion of overlapping linguistic categories is introduced to handle situations for which the objective stated in Section 3.1, whereby the linguistic category of a Frogans address should be clearly distinguishable by humans or, failing that, by systems, cannot be fully achieved.

As stated in Section 5, the validity of a network name associated with a linguistic category depends not only on the employable characters of that linguistic category, but also on its arrangement rules.

In order to determine whether a linguistic category, called LC, overlaps with other linguistic categories, the following types of employable characters of LC and, when applicable, arrangement rules need to be taken into account:

A. Connector characters.

Some connector characters that are employable characters of LC can also be employable characters of other linguistic categories.

The IFAP specification [IFAP] defines rules concerning the use of connector characters in network names (see IFAP, section 4.4 Connector Characters).

As a result of those rules, the mere fact that these characters are also employable characters of other linguistic categories does not cause LC to overlap with these linguistic categories.

B. Decimal digits that are characters with the General Category of Nd (Decimal_Number), as defined in the Unicode Standard [Unicode] (see the Unicode Standard, section 4.5 General Category).

Some of these characters that are employable characters of LC can also be employable characters of other linguistic categories.

The IFAP specification defines rules concerning the use of decimal numbers in network names (see IFAP, section 4.2 Network Name).

As a result of those rules, the mere fact that these characters are also employable characters of other linguistic categories does not cause LC to overlap with these linguistic categories.

C. Characters borrowed from a writing system corresponding to another linguistic category.

Some characters that are employable characters of LC can be borrowed from a writing system corresponding to another linguistic category.

The mere fact that these characters are also employable characters of other linguistic categories could cause LC to overlap with these linguistic categories. In order to prevent this situation, it is necessary to define an arrangement rule of LC stating that the network name contains at least one character included in one of the writing systems corresponding to LC.

D. Characters from one of the writing systems corresponding to LC that are borrowed by other linguistic categories.

Some characters that are employable characters of LC and that are from one of the writing systems corresponding to LC can be borrowed by other linguistic categories.

The mere fact that these characters are also employable characters of other linguistic categories could cause LC to overlap with these linguistic categories. In order to prevent this situation, for each of these linguistic categories, it is necessary to define an arrangement rule stating that the network name contains at least one character included in one of the writing systems corresponding to that linguistic category.

E. Characters with the Han Unicode Script property [UAX24].

Some of these characters that are employable characters of LC can also be employable characters of other linguistic categories.

The mere fact that some of these characters are also employable characters of other linguistic categories causes LC to overlap with these linguistic categories.

Such characters are used in the writing systems of three of the world's major languages: the Chinese, Japanese, and Korean languages [Unicode] (see the Unicode Standard, appendix E Han Unification History).

7. Generating Convergence Forms

This section provides methods used to generate Intra-LC convergence forms and Inter-LC convergence forms.

The Intra-LC convergence form of each type for a valid network name or a valid site name associated with a linguistic category is generated using the preferred form of that network name or site name. The Inter-LC convergence form for a valid network name is generated using the preferred form of that network name.

Intra-LC convergence forms and Inter-LC convergence forms are not generated using the reference form of a network name or a site name defined in the IFAP specification [IFAP]. Reference forms are designed for other purposes such as evaluating the length of a network name or a site name, and checking whether two network names or site names are identical. Moreover, unlike preferred forms, reference forms are not intended for display to end users.

In this version of FACR, all Intra-LC convergence forms and Inter-LC convergence forms are strings of Unicode characters [Unicode]. Other formats of convergence forms may be introduced in future versions of FACR.

The strings of Unicode characters that represent Intra-LC convergence forms and Inter-LC convergence forms do not necessarily comply with the IFAP specification.

7.1. Intra-LC Convergence Forms

As stated in Section 3.2, there can be more than one type of Intra-LC convergence form that apply to the network names and site names associated with a linguistic category.

Each Intra-LC convergence form type is identified using a unique label characterizing the type. The label is a string of ASCII characters [ASCII] starting with the six characters 'I' (0x49), 'n' (0x6E), 't' (0x74), 'r' (0x72), 'a' (0x61) and '-' (0x2D), followed by the label of the linguistic category, followed by '-' (0x2D), followed by between 3 and 16 characters from 'A' to 'Z' (0x41-0x5A) or from 'a' to 'z' (0x61-0x7A).

One Intra-LC convergence form type of a linguistic category can be defined using the Unicode Technical Standard #39 [UTS39] (see the Unicode Technical Standard #39, section 4 Confusable Detection) as a source, in accordance with Section 3.2.

The identifier of that Intra-LC convergence form type of a linguistic category ends with '-' (0x2D) followed by 'C' (0x43), 'o' (0x6F), 'n' (0x6E), 'f' (0x66), 'u' (0x75), 's' (0x73), 'a' (0x61), 'b' (0x62), 'l' (0x6C) and 'e' (0x65).

The Intra-LC convergence form of that type for a valid network name or a valid site name associated with a linguistic category is the string of Unicode characters [Unicode] generated by applying to the preferred form of the network name or the site name the skeleton(X) transform described in the Unicode Technical Standard #39. The specified data table used in the skeleton(X) transform is the Mixed-Script Any-Case (MA) table, which is adapted using the method described below in order to make the transform compatible with both the Frogans address pattern defined in the IFAP specification [IFAP] and the employable characters of the linguistic category.

Despite the adaptation of the MA table, the transform remains idempotent, and therefore there is no need to apply it recursively.

The Unicode Technical Report #36 [UTR36], which focuses on visual and non-visual security issues, states that users expect diacritical marks (such as an accent, a tone, or some other linguistic information) to distinguish domain names (see the Unicode Technical Report #36, section 2.1 Internationalized Domain Names). This principle is respected in the skeleton(X) transform described in the Unicode Technical Standard #39.

As a result, the Intra-LC convergence form of this type is different for two network names or two site names that only differ by a character having a diacritical mark in one network name or site name but not in the other. For example, the convergence form of that type for a network name containing a U+006E LATIN SMALL LETTER N character is different from the convergence form of the same type for another network name where that character is replaced by the U+00F1 LATIN SMALL LETTER N WITH TILDE character.

If other Intra-LC convergence form types are needed for a linguistic category, they can be defined in accordance with Section 3.2.

For assistance in implementing a function to generate Intra-LC convergence forms, see Appendix C.3.

The method used to adapt the MA table takes the following value as input:

* LC: the linguistic category

The method uses the following terms and conventions:

* mapping table: a table with two columns. For each row in the mapping table, the first column contains a source code point, and the second column contains an array of code points that the code point in the first column is mapped to.

* 1-to-0 mapping: a row in a mapping table where the source code point is mapped to an array of code points that is empty.

* 1-to-1 mapping: a row in a mapping table where the source code point is mapped to an array of code points containing a single code point.

* 1-to-n mapping: a row in a mapping table where the source code point is mapped to an array of code points containing more than one code point.

* mapping function: a function defined for a mapping table that searches for an input code point in the first column of that mapping table and, if found, returns the array of code points in the second column of the same row. If the input code point is not found, then the function returns the input code point, indicating that the code point is mapped to itself.

The method uses the following preprocessed data:

* CM table: a mapping table customized to handle particular cases in the method, where:

  - U+0181 is mapped to: U+0062, U+0027
  - U+018A is mapped to: U+0064, U+0027
  - U+01A4 is mapped to: U+0070, U+0027
  - U+01AC is mapped to: U+0074, U+0027
  - U+01B3 is mapped to: U+0079, U+0027
  - U+0256 is mapped to: U+0044, U+0335
  - U+0314 is mapped to: U+0027
  - U+0629 is mapped to: U+006F, U+0308
  - U+2202 is mapped to: U+0044

* M1 table: a mapping table containing the mappings defined in the MA table, where:

  - The first column contains all the code points for which a mapping is defined in the MA table.

  - The second column contains the code points to which the code points in the first column are mapped in the MA table.

* MM table: a mapping table containing a modified copy of the M1 table, which integrates the mappings defined in the CM table, where:

  - All the rows in the M1 table are copied to the MM table, except for those in which the source code point is a source code point in the CM table.

  - All the rows in the CM table are added to the resulting MM table.

  - Each code point, called CP, found in any array of code points in the resulting MM table that is a source code point in the CM table is replaced in that array by the array of code points to which CP is mapped in the CM table.

* KC table: a mapping table used to take into account the caseless matching process defined in the IFAP specification for generating the reference form, where:

  - The first column contains all the code points corresponding to the employable characters of LC.

  - The second column contains the code points corresponding to the NFKC_Casefold derived property of the code point in the first column, where the NFKC_Casefold derived property is defined in the Unicode Standard Annex #44 [UAX44] (see the Unicode Standard Annex #44, section 5.3 Property Definitions).

* L1 list: a list of code points corresponding to all the eligible characters defined in the IFAP specification.

* L2 list: a list of code points resulting from the NFD decomposition of each code point in the L1 list, where the NFD decomposition is defined in the Unicode Standard [Unicode] (see the Unicode Standard, section 3.11 Normalization Forms, definition D118).

* L3 list: a list of code points corresponding to the NFKC_Casefold derived property of the code points in the L1 and L2 lists, where the NFKC_Casefold derived property is defined in the Unicode Standard Annex #44 [UAX44] (see the Unicode Standard Annex #44, section 5.3 Property Definitions).

* L4 list: a list of code points resulting from applying the mapping function defined for the MM table to each of the code points in the L1 and L2 lists.

- LM list: a list of code points resulting from the merging of the code points in the L1, L2, L3, and L4 lists.

- TT list: a list of code points containing the source code points corresponding to the 1-to-1 mappings in the MM table or in the KC table, as well as the single code points to which the source code points are mapped. Duplicates of a code point are removed from the list.

The method to adapt the MA table produces the following output:

- AD table: the adapted table. It is a mapping table.

The method consists of performing the following steps:

1. All the code points in the TT list are divided into code point classes. A code point class contains code points that are directly or indirectly linked, according to the following definitions:

   a) Two code points in the TT list, called CPA and CPB, are directly linked if CPA is mapped to CPB or if CPB is mapped to CPA in a 1-to-1 mapping in either the MM table or the KC table.

   b) Two code points in the TT list, called CPA and CPB, are indirectly linked if there exists at least one series of N code points in the TT list, called CP[1] to CP[N], where all the following conditions are met:

      - if N equals one, then:

        - CPA and CP[1] are directly linked
        - CP[1] and CPB are directly linked

      - if N is greater than one, then:

        - CPA and CP[1] are directly linked
        - for each K between 1 and (N - 1), CP[K] and CP[K+1] are directly linked
        - CP[N] and CPB are directly linked

   As a result, each code point found in the TT list is included in only one code point class.

   Furthermore, each code point in a class is neither directly nor indirectly linked to a code point in any other class.

It should be noted that the fact that two code points in a class are indirectly linked does not exclude the possibility that these code points are also directly linked.

2. In each code point class, the code points that are not included in the LM list are removed from that class. In the remainder of the method, code point classes that contain only one code point are ignored.

3. A mapping table, called A1 table, is created. For each code point class:

   i. If no code point in the class is the source code point of a 1-to-n mapping defined in the MM table or in the KC table, then:

      - The code point in the class with the lowest value, called CPX, is found.

      - For each code point in the class, called CP, that is different from CPX, a 1-to-1 mapping is added to the A1 table where the source code point is CP and the array of code points contains CPX only.

   ii. Otherwise, if there is only one code point in the class, called CPY, that is the source code point of a 1-to-n mapping defined in the MM table or in the KC table, then:

      - The array of code points, called ARY, to which CPY is mapped in that 1-to-n mapping, is retrieved.

      - For each code point in the class, called CP, a 1-to-n mapping is added to the A1 table where the source code point is CP and the array of code points is ARY.

   iii. Otherwise, if there is a series of N code points in the class, called CPY[1] to CPY[N], where N is greater than one and each code point in the series is the source code point of a 1-to-n mapping defined in the MM table or in the KC table, then:

      - The code point in the series of N code points with the lowest value is found, along with its index, called KX. The value of KX is between 1 and N.

      - The array of code points, called ARY[KX], to which CPY[KX] is mapped in those 1-to-n mappings, is retrieved.

- For each code point in the class, called CP, a 1-to-n mapping is added to the A1 table where the source code point is CP and the array of code points is ARY[KX].

4. Each code point, called CP, in the LM list which is not in any code point class is processed as follows:

   i. If CP is the source code point of a 1-to-n mapping defined in the MM table, then that 1-to-n mapping is added to the A1 table.

   ii. Otherwise, if CP is the source code point of a 1-to-n mapping defined in the KC table, then that 1-to-n mapping is added to the A1 table.

5. In this final step of the method, the AD table is copied from the A1 table using the process hereafter.

   For each row in the A1 table, called R1:

   i. An array of code points, called ARD, is created as follows:

      - The array of code points in R1 is copied to ARD.

      - The mapping function defined for the A1 table is applied recursively to each code point found in ARD until there are no changes in ARD.

   ii. A row is added to the AD table, where:

      - The source code point is the source code point in R1.

      - The array of code points is ARD.

   This process ensures that transforms using the AD table are idempotent.

7.2. Inter-LC Convergence Form

As stated in Section 3.2, Inter-LC convergence forms do not apply to site names and there is only one type of Inter-LC convergence form that applies to network names, regardless of the linguistic category with which they are associated.

The Inter-LC convergence form type is defined using the Unicode Technical Standard #39 [UTS39] (see the Unicode Technical Standard #39, section 4 Confusable Detection) as a source, in accordance with Section 3.2.

The Inter-LC convergence form type is identified using a unique label. The label is a string of ASCII characters [ASCII] consisting of the eight characters 'I' (0x49), 'n' (0x6E), 't' (0x74), 'e' (0x65), 'r' (0x72), '-' (0x2D), 'L' (0x4C) and 'C' (0x43).

The Inter-LC convergence form for a valid network name is the string of Unicode characters [Unicode] generated by applying to the preferred form of the network name the skeleton(X) transform described in the Unicode Technical Standard #39. The specified data table used in the skeleton(X) transform is the Mixed-Script Any-Case (MA) table, which is adapted using the method described below in order to make the transform compatible with the Frogans address pattern defined in the IFAP specification [IFAP].

Despite the adaptation of the MA table, the transform remains idempotent, and therefore there is no need to apply it recursively.

The Unicode Technical Report #36 [UTR36], which focuses on visual and non-visual security issues, states that users expect diacritical marks (such as an accent, a tone, or some other linguistic information) to distinguish domain names (see the Unicode Technical Report #36, section 2.1 Internationalized Domain Names). This principle is respected in the skeleton(X) transform described in the Unicode Technical Standard #39.

As a result, the Inter-LC convergence form is different for two network names that only differ by a character having a diacritical mark in one network name but not in the other. For example, the convergence form of that type for a network name containing a U+006E LATIN SMALL LETTER N character is different from the convergence form of the same type for another network name where that character is replaced by the U+00F1 LATIN SMALL LETTER N WITH TILDE character.

For assistance in implementing a function to generate Inter-LC convergence forms, see Appendix C.4.

The method used to adapt the MA table is the method used to adapt the MA table described in Section 7.1, modified by the following changes:

* No linguistic category is specified as input.

* In the KC table, the employable characters in the first column are replaced by all the eligible characters defined in the IFAP specification.

* Two 1-to-0 mappings are added to the A1 table. The characters corresponding to the source code points in these 1-to-0 mappings are:

  - U+200C ZERO WIDTH NON-JOINER
  - U+200D ZERO WIDTH JOINER

  As a result, in the mapping of the skeleton(X) transform, such characters are filtered out of the target string.

8. Checking Whether Two Valid Network Names are Convergent

This section describes the method for checking whether two valid network names are convergent. It is assumed in this section that each of the two valid network names is associated with a linguistic category.

The method takes the following values as input:

* LC1: the first linguistic category

* LC2: the second linguistic category

* NN1: the valid network name for LC1

* NN2: the valid network name for LC2

The method consists of performing the following tests in succession until it has been determined whether or not NN1 and NN2 are convergent:

A. If NN1 and NN2 are identical according to the IFAP specification [IFAP], then NN1 and NN2 are convergent, irrespective of LC1 and LC2.

B. If LC1 and LC2 are the same linguistic category, called LC, then two cases can arise:

   1. If there is an Intra-LC convergence form type of LC where the Intra-LC convergence form values of NN1 and NN2 are the same, then NN1 and NN2 are convergent.

   2. Otherwise, two further cases can arise:

      i. If there is a linguistic category, called LCo, that overlaps with LC where NN1 and NN2 are valid network names for LCo,

         and

         there is an Intra-LC convergence form type of LCo where the Intra-LC convergence form values of NN1 and NN2 are the same,

         then NN1 and NN2 are convergent.

ii. Otherwise, NN1 and NN2 are not convergent.

C. If LC1 and LC2 are different linguistic categories, then two cases can arise:

1. If LC1 does not overlap with LC2, then three further cases can arise:

i. If there is a linguistic category different from LC1 and LC2, called LCo, that overlaps with LC1 and LC2 where NN1 and NN2 are valid network names for LCo,

and

there is an Intra-LC convergence form type of LCo where the Intra-LC convergence form values of NN1 and NN2 are the same,

then NN1 and NN2 are convergent.

ii. Otherwise, if the Inter-LC convergence form values of NN1 and NN2 are the same, then NN1 and NN2 are convergent.

iii. Otherwise, NN1 and NN2 are not convergent.

2. If LC1 overlaps with LC2, then five further cases can arise:

i. If NN2 is a valid network name for LC1,

and

there is an Intra-LC convergence form type of LC1 where the Intra-LC convergence form values of NN1 and NN2 are the same,

then NN1 and NN2 are convergent.

ii. Otherwise, if NN1 is a valid network name for LC2,

and

there is an Intra-LC convergence form type of LC2 where the Intra-LC convergence form values of NN1 and NN2 are the same,

then NN1 and NN2 are convergent.

iii. Otherwise, if there is a linguistic category different from LC1 and LC2, called LCo, that overlaps with LC1 and LC2 where NN1 and NN2 are valid network names for LCo,

and

there is an Intra-LC convergence form type of LCo where the Intra-LC convergence form values of NN1 and NN2 are the same,

then NN1 and NN2 are convergent.

iv. Otherwise, if the Inter-LC convergence form values of NN1 and NN2 are the same, then NN1 and NN2 are convergent.

v. Otherwise, NN1 and NN2 are not convergent.

9. Checking Whether Two Valid Site Names are Convergent

This section describes the method for checking whether two valid site names are convergent. It is assumed in this section that the two valid site names are used with a common valid network name that is associated with a linguistic category.

The method does not check the convergence of two valid site names that are used with different valid network names or that are associated with different linguistic categories.

The method takes the following values as input:

- LC: the linguistic category

- NN: the common valid network name for LC

- SN1: the first valid site name used with NN

- SN2: the second valid site name used with NN

The method consists of performing the following tests in succession until it has been determined whether or not SN1 and SN2 are convergent:

A. If SN1 and SN2 are identical according to the IFAP specification [IFAP], then SN1 and SN2 are convergent.

B. Otherwise, two cases can arise:

1. If there is an Intra-LC convergence form type of LC where the Intra-LC convergence form values of SN1 and SN2 are the same, then SN1 and SN2 are convergent.

2. Otherwise, two further cases can arise:

   i. If there is a linguistic category, called LCo, that overlaps with LC where NN is a valid network name for LCo, and where SN1 and SN2 used with NN are valid site names for LCo,

   and

   there is an Intra-LC convergence form type of LCo where the Intra-LC convergence form values of SN1 and SN2 are the same,

   then SN1 and SN2 are convergent.

ii. Otherwise, SN1 and SN2 are not convergent.

10. Available Linguistic Categories

This section describes the linguistic categories with which network names and site names can be associated.

Since Frogans addresses are designed to be used worldwide, the objective of FACR is that, within the possibilities offered by the International Frogans Address Pattern (IFAP) specification [IFAP], every language or writing system corresponds to a linguistic category.

Processing the vast number of languages and the diversity of writing systems in question is a significant task which unfortunately cannot be fully achieved in this first version of FACR. Priority has to be given to completing the task of establishing a flexible and modular architecture for FACR so as to make the specification easy to upgrade.

In order to define an inclusive set of available linguistic categories for this first version of FACR, the popularity of languages and writing systems in online publishing is used as the guiding principle. Given that there is no single widely-recognized and uncontested source for determining this popularity, various sources of information related to online publishing are used. This information includes, for example, the number of registered country-code Top-Level Domains (ccTLDs), the number of ICANN-accredited registrars per country, the scripts appearing in the Trademark Clearinghouse introduced by ICANN's new generic Top-Level Domain (gTLD) program, and statistics on Internet usage such as the number of users or page views by language.

After analyzing and processing this information, 10 available linguistic categories are defined for this version of FACR, in accordance with the rules stated in Section 3.1:

LC-Latin, LC-Chinese, LC-Japanese, LC-Korean, LC-Arabic, LC-Cyrillic, LC-Hebrew, LC-Devanagari, LC-Thai, and LC-Greek.

Each available linguistic category is described in a separate section which defines the corresponding languages and writing systems, the employable characters, the arrangement rules, the overlapping linguistic categories, and the Intra-LC convergence form types.

For information concerning the addition of new linguistic categories in future versions of FACR, see Section 11.

10.1. LC-Latin

The label 'LC-Latin' is the identifier of the Latin linguistic category, which corresponds to a group of languages that use the same writing system.

This section defines, for LC-Latin, the corresponding languages and writing system, the employable characters, the arrangement rules, the overlapping linguistic categories, and the Intra-LC convergence form types.

10.1.1. Languages and Writing System Corresponding to LC-Latin

The following rules are defined in accordance with the rules stated in Section 3.1.

The writing system corresponding to LC-Latin is the Latin writing system.

The languages corresponding to LC-Latin are the languages listed in the Unicode Common Locale Data Repository (CLDR) [CLDR] for which the <language> element meets the following conditions:

- the value of the "scripts" attribute contains 'Latn', and

- the value of the "alt" attribute is not equal to 'secondary'

As a result, a total of 457 languages, including territorial variations, correspond to LC-Latin. These languages comprise, among others, English, Spanish, Portuguese, French, Indonesian, Swahili, German, Javanese, Vietnamese, Turkish, Filipino, Italian, and Polish.

10.1.2. Employable Characters of LC-Latin

In accordance with the rules stated in Section 4.1, the primary source used for determining the employable characters of LC-Latin is either an IDN table included in the Repository of Internationalized Domain Name (IDN) Practices maintained by the Internet Assigned Numbers Authority (IANA) [IANA-Repository], or data included in the Unicode Common Locale Data Repository (CLDR) maintained by the Unicode Consortium [CLDR].

In the Repository of IDN Practices, there is no IDN table that contains all the characters commonly used in the languages corresponding to LC-Latin.

Therefore, for LC-Latin, no IDN table meets all the requirements in Section 4.1.

In accordance with the rules stated in Section 4.1, the primary source used for determining the employable characters of LC-Latin is data included in CLDR.

A character is an employable character of LC-Latin if all the following conditions are met:

1. The character is an eligible character according to the International Frogans Address Pattern specification (IFAP) [IFAP].

2. The character does not correspond to any of the following code points:

   * U+002A. This code point corresponds to the separator character in a Frogans address.

   * U+02BB, U+02BC. These code points correspond to characters that are visually confusable with the U+0027 APOSTROPHE character.

   * U+0300, U+0301, U+0302, U+0303, U+0304, U+0308, U+030C, U+0327, U+0331, U+1DC6, U+1DC7. These code points correspond to combining marks.

3. The character is accepted as a potential employable character by the method described in Section 4.1.2 where the script subtag is 'Latn' and the option to include auxiliary exemplar sets is disabled.

As a result, the only connector characters that can be used in a network name or a site name associated with LC-Latin are the U+002D HYPHEN-MINUS and the U+00B7 MIDDLE DOT characters.

Furthermore, the decimal digits that can be used in a network name or a site name associated with LC-Latin are the 10 decimal digits in the range from U+0030 DIGIT ZERO to U+0039 DIGIT NINE.

In this version of FACR, LC-Latin has a total of 421 employable characters. According to the data in Unicode CLDR, these employable characters can be used with at least 145 of the 457 languages corresponding to LC-Latin.

For assistance in implementing a process that verifies whether a candidate string corresponding to a network name or a site name associated with a linguistic category complies with the employable character rules defined in this FACR specification, see Appendix C.1.

10.1.3. Arrangement Rules of LC-Latin

The following rules are defined in accordance with the rules stated in Section 4.2.

The arrangement rules applicable to a network name associated with LC-Latin are:

* The network name can contain only one type of connector character.

* If the type of connector character in the network name is the U+00B7 MIDDLE DOT character, then each occurrence of that connector character is immediately preceded and followed by the U+004C LATIN CAPITAL LETTER L character or is immediately preceded and followed by the U+006C LATIN SMALL LETTER L character.

  This arrangement rule is inspired by the rules that describe the contexts in which particular characters are permitted, defined in RFC 5892 [RFC5892], which is part of Internationalized Domain Names for Applications [IDNA2008] (see RFC 5892, appendix A.3 MIDDLE DOT).

The arrangement rules applicable to a site name associated with LC-Latin are:

* If the network name contains a connector character, then the site name cannot contain a different type of connector character.

* If the network name does not contain a connector character, then the site name can contain only one type of connector character.

* The preceding arrangement rule applicable to the network name concerning the U+00B7 MIDDLE DOT character also applies to the site name.

The preceding arrangement rules concerning connector characters complement the rules concerning connector characters defined in the International Frogans Address Pattern (IFAP) specification [IFAP] (see IFAP, section 4.4. Connector Characters).

These arrangement rules of LC-Latin for a site name have the same outcome irrespective of the preferred form of the network name, as required by Section 4.2.

For assistance in implementing a process that verifies whether a candidate string corresponding to a network name or a site name associated with a linguistic category complies with the arrangement rules defined in this FACR specification, see Appendix C.2.

10.1.4. Linguistic Categories Overlapping With LC-Latin

In accordance with the rules stated in Section 6, LC-Latin overlaps with another linguistic category if LC-Latin has valid network names in common with that linguistic category. The validity of a network name associated with a linguistic category depends not only on the employable characters of that linguistic category, but also on its arrangement rules.

In order to determine whether LC-Latin overlaps with other linguistic categories, the following employable characters and, when applicable, arrangement rules are taken into account:

A. The connector characters U+002D HYPHEN-MINUS and U+00B7 MIDDLE DOT.

These characters are also employable characters of other linguistic categories.

In accordance with Section 6, the mere fact that these characters are also employable characters of other linguistic categories does not cause LC-Latin to overlap with these linguistic categories.

B. The 10 decimal digits in the range from U+0030 DIGIT ZERO to U+0039 DIGIT NINE.

These characters are also employable characters of other linguistic categories.

In accordance with Section 6, the mere fact that these characters are also employable characters of other linguistic categories does not cause LC-Latin to overlap with these linguistic categories.

C. The characters borrowed from a writing system corresponding to another linguistic category.

No employable characters of LC-Latin are characters borrowed from a writing system corresponding to another linguistic category.

D. The characters from the writing system corresponding to LC-Latin that are borrowed by other linguistic categories.

These are characters borrowed by LC-Chinese, LC-Japanese, LC-Korean, and LC-Thai. They are the 52 uppercase and lowercase alphabetical characters in the ranges from the U+0041 LATIN CAPITAL LETTER A character to the U+005A LATIN CAPITAL LETTER Z character (inclusive) and from the U+0061 LATIN SMALL LETTER A character to the U+007A LATIN SMALL LETTER Z character (inclusive).

In accordance with Section 6, and given the arrangement rules of LC-Chinese, LC-Japanese, LC-Korean, and LC-Thai, the mere fact that these characters are also employable characters of LC-Chinese, LC-Japanese, LC-Korean, and LC-Thai does not cause LC-Latin to overlap with these linguistic categories.

E. The characters with the Han Unicode Script property [UAX24].

No employable characters of LC-Latin are characters with the Han Unicode Script property.

As a result, LC-Latin does not overlap with any other linguistic category.

10.1.5. Intra-LC Convergence Form Types of LC-Latin

The following rules are defined in accordance with the rules stated in Section 3.2 and Section 7.1.

There is only one type of Intra-LC convergence form that applies to network names and site names associated with LC-Latin.

The Intra-LC convergence form type is defined using the Unicode Technical Standard #39 [UTS39] (see the Unicode Technical Standard #39, section 4 Confusable Detection) as a source, in accordance with Section 3.2.

The identifier of this Intra-LC convergence form type is the label 'Intra-LC-Latin-Confusable'.

The Intra-LC convergence form of this type for a valid network name or a valid site name associated with LC-Latin is the string of Unicode characters [Unicode] generated according to the rules stated in Section 7.1.

For assistance in implementing a function to generate Intra-LC convergence forms, see Appendix C.3.

10.2. LC-Chinese

The label 'LC-Chinese' is the identifier of the Chinese linguistic category, which corresponds to a language that uses two writing systems.

This section defines, for LC-Chinese, the corresponding language and writing systems, the employable characters, the arrangement rules, the overlapping linguistic categories, and the Intra-LC convergence form types.

10.2.1. Language and Writing Systems Corresponding to LC-Chinese

The following rules are defined in accordance with the rules stated in Section 3.1.

The language corresponding to LC-Chinese is the Chinese language.

The writing systems corresponding to LC-Chinese are:

* Traditional Chinese

* Simplified Chinese

10.2.2. Employable Characters of LC-Chinese

In accordance with the rules stated in Section 4.1, the primary source used for determining the employable characters of LC-Chinese is either an IDN table included in the Repository of Internationalized Domain Name (IDN) Practices maintained by the Internet Assigned Numbers Authority (IANA) [IANA-Repository], or data included in the Unicode Common Locale Data Repository (CLDR) maintained by the Unicode Consortium [CLDR].

In the Repository of IDN Practices, there is an IDN table for .cn [IDN-CN], the country-code Top-Level Domain (ccTLD) for the People's Republic of China. This IDN table is used for the registration of IDNs in the Chinese language. It contains characters used in the Simplified Chinese writing system and in the Traditional Chinese writing system.

In the Repository of IDN Practices, there is also an IDN table for .tw [IDN-TW], the ccTLD for Taiwan. This IDN table is also used for the registration of IDNs in the Chinese language. The characters that are permitted in this IDN table are exactly the same as those permitted in the IDN table for the .cn ccTLD.

Therefore, for LC-Chinese, it is considered that the IDN table for the .cn ccTLD meets all the requirements in Section 4.1.

In accordance with the rules stated in Section 4.1, the primary source used for determining the employable characters of LC-Chinese is the IDN table for the .cn ccTLD.

A character is an employable character of LC-Chinese if all the following conditions are met:

1. The character is an eligible character according to the International Frogans Address Pattern specification (IFAP) [IFAP].

2. The character does not correspond to the following code point: U+002A. This code point corresponds to the separator character in a Frogans address.

3. The character is accepted as a potential employable character by the method described in Section 4.1.1 where the IDN table is the IDN table for the .cn ccTLD.

As a result, the only connector character that can be used in a network name or a site name associated with LC-Chinese is the U+002D HYPHEN-MINUS character.

Furthermore, the decimal digits that can be used in a network name or a site name associated with LC-Chinese are the 10 decimal digits in the range from U+0030 DIGIT ZERO to U+0039 DIGIT NINE.

In addition, the following characters from the Latin writing system are employable characters of LC-Chinese: the ranges from the U+0041 LATIN CAPITAL LETTER A character to the U+005A LATIN CAPITAL LETTER Z character (inclusive), and from the U+0061 LATIN SMALL LETTER A character to the U+007A LATIN SMALL LETTER Z character (inclusive).

In this version of FACR, LC-Chinese has a total of 19,583 employable characters.

For assistance in implementing a process that verifies whether a candidate string corresponding to a network name or a site name associated with a linguistic category complies with the employable character rules defined in this FACR specification, see Appendix C.1.

10.2.3. Arrangement Rules of LC-Chinese

The following rules are defined in accordance with the rules stated in Section 4.2.

The arrangement rules applicable to a network name associated with LC-Chinese are:

* The network name contains at least one character with the Han Unicode Script property [UAX24].

  This arrangement rule is required by Section 6, since LC-Chinese includes characters borrowed from the writing system corresponding to LC-Latin.

There are no arrangement rules applicable to a site name associated with LC-Chinese.

For assistance in implementing a process that verifies whether a candidate string corresponding to a network name or a site name associated with a linguistic category complies with the arrangement rules defined in this FACR specification, see Appendix C.2.

10.2.4. Linguistic Categories Overlapping With LC-Chinese

In accordance with the rules stated in Section 6, LC-Chinese overlaps with another linguistic category if LC-Chinese has valid network names in common with that linguistic category. The validity of a network name associated with a linguistic category depends not only on the employable characters of that linguistic category, but also on its arrangement rules.

In order to determine whether LC-Chinese overlaps with other linguistic categories, the following employable characters and, when applicable, arrangement rules are taken into account:

A. The connector character U+002D HYPHEN-MINUS.

   This character is also an employable character of other linguistic categories.

   In accordance with Section 6, the mere fact that this character is also an employable character of other linguistic categories does not cause LC-Chinese to overlap with these linguistic categories.

B. The 10 decimal digits in the range from U+0030 DIGIT ZERO to U+0039 DIGIT NINE.

   These characters are also employable characters of other linguistic categories.

   In accordance with Section 6, the mere fact that these characters are also employable characters of other linguistic categories does not cause LC-Chinese to overlap with these linguistic categories.

C. The characters borrowed from a writing system corresponding to another linguistic category.

These are characters borrowed from the writing system corresponding to LC-Latin. They are the 52 uppercase and lowercase alphabetical characters in the ranges from the U+0041 LATIN CAPITAL LETTER A character to the U+005A LATIN CAPITAL LETTER Z character (inclusive) and from the U+0061 LATIN SMALL LETTER A character to the U+007A LATIN SMALL LETTER Z character (inclusive).

These characters are also included in the employable characters of LC-Japanese, LC-Korean, and LC-Thai.

In accordance with Section 6, and given the arrangement rules of LC-Chinese, the mere fact that these characters are also employable characters of LC-Latin, LC-Japanese, LC-Korean, and LC-Thai does not cause LC-Chinese to overlap with these linguistic categories.

D. The characters from the writing systems corresponding to LC-Chinese that are borrowed by other linguistic categories.

With the exception of characters with the Han Unicode Script property, no employable characters of LC-Chinese that are also characters from one of the writing systems corresponding to LC-Chinese are borrowed by another linguistic category.

E. The characters with the Han Unicode Script property.

There are 19,520 of these characters, out of which:

- 6,181 are also employable characters of LC-Japanese
- 752 are also employable characters of LC-Korean

In accordance with Section 6, the mere fact that some of these characters are also employable characters of LC-Japanese and LC-Korean causes LC-Chinese to overlap with these linguistic categories.

As a result, LC-Chinese overlaps with the following linguistic categories: LC-Japanese and LC-Korean.

10.2.5. Intra-LC Convergence Form Types of LC-Chinese

The following rules are defined in accordance with the rules stated in Section 3.2 and Section 7.1.

There are two types of Intra-LC convergence forms that apply to network names and site names associated with LC-Chinese.

The first Intra-LC convergence form type is defined using the Unicode Technical Standard #39 [UTS39] (see the Unicode Technical Standard #39, section 4 Confusable Detection) as a source, in accordance with Section 3.2.

The identifier of this first Intra-LC convergence form type is the label 'Intra-LC-Chinese-Confusable'.

The Intra-LC convergence form of this first type for a valid network name or a valid site name associated with LC-Chinese is the string of Unicode characters [Unicode] generated according to the rules stated in Section 7.1.

The second Intra-LC convergence form type is defined using RFC 3743 [RFC3743] and IDN tables included in the Repository of Internationalized Domain Name (IDN) Practices maintained by the Internet Assigned Numbers Authority (IANA) [IANA-Repository] as a source, in accordance with Section 3.2.

The identifier of this second Intra-LC convergence form type is the label 'Intra-LC-Chinese-Variant'.

The Intra-LC convergence form of this second type for a valid network name or a valid site name associated with LC-Chinese is the string of Unicode characters generated by applying to the preferred form of the network name or the site name the following transform:

* Each character in the input string is mapped successively to a character in the target string according to the variant mapping table defined below.

* If a character in the input string is not found in the variant mapping table, the character is mapped to itself in the target string.

The transform is idempotent, and therefore there is no need to apply it recursively.

The variant mapping table has two columns:

- The first column contains a source code point corresponding to an employable character of LC-Chinese.

- The second column contains a code point to which the employable character in the first column is mapped.

The rows of the variant mapping table are defined using the method hereafter.

For each employable character of LC-Chinese, whose corresponding code point is called CP, a set is defined, comprising:

- the code points corresponding to the Preferred Variant and to the Character Variants of CP in the IDN table for .cn ccTLD [IDN-CN]

- the code points corresponding to the Preferred Variant and to the Character Variants of CP in the IDN table for .tw ccTLD [IDN-TW]

- the code point corresponding to the NFKC_Casefold derived property of CP, where the NFKC_Casefold derived property is defined in the Unicode Standard Annex #44 [UAX44] (see the Unicode Standard Annex #44, section 5.3 Property Definitions)

This set is not empty and can contain several identical point codes.

A row is defined for CP in the variant mapping table if CP is greater than the smallest code point in the set, called CPX, in which case the first column in the row contains CP and the second column contains CPX.

For assistance in implementing a function to generate Intra-LC convergence forms, see Appendix C.3.

10.3. LC-Japanese

The label 'LC-Japanese' is the identifier of the Japanese linguistic category, which corresponds to a language that uses three writing systems.

This section defines, for LC-Japanese, the corresponding language and writing systems, the employable characters, the arrangement rules, the overlapping linguistic categories, and the Intra-LC convergence form types.

10.3.1. Language and Writing Systems Corresponding to LC-Japanese

The following rules are defined in accordance with the rules stated in Section 3.1.

The language corresponding to LC-Japanese is the Japanese language.

The writing systems corresponding to LC-Japanese are:

* Hiragana

* Katakana

* Kanji

Kanji corresponds to characters of Chinese origin.

10.3.2. Employable Characters of LC-Japanese

In accordance with the rules stated in Section 4.1, the primary source used for determining the employable characters of LC-Japanese is either an IDN table included in the Repository of Internationalized Domain Name (IDN) Practices maintained by the Internet Assigned Numbers Authority (IANA) [IANA-Repository], or data included in the Unicode Common Locale Data Repository (CLDR) maintained by the Unicode Consortium [CLDR].

In the Repository of IDN Practices, there is an IDN table for .jp [IDN-JP], the country-code Top-Level Domain (ccTLD) for Japan. This IDN table is used for the registration of IDNs in the Japanese language. It contains characters used in the Hiragana, Katakana, and Kanji writing systems.

Therefore, for LC-Japanese, it is considered that the IDN table for the .jp ccTLD meets all the requirements in Section 4.1.

In accordance with the rules stated in Section 4.1, the primary source used for determining the employable characters of LC-Japanese is the IDN table for the .jp ccTLD.

A character is an employable character of LC-Japanese if all the following conditions are met:

1. The character is an eligible character according to the International Frogans Address Pattern specification (IFAP) [IFAP].

2. The character does not correspond to the following code point: U+002A. This code point corresponds to the separator character in a Frogans address.

3. The character is accepted as a potential employable character by the method described in Section 4.1.1 where the IDN table is the IDN table for the .jp ccTLD.

As a result, the connector characters that can be used in a network name or a site name associated with LC-Japanese are the U+002D HYPHEN-MINUS character and the U+30FB KATAKANA MIDDLE DOT character.

Furthermore, the decimal digits that can be used in a network name or a site name associated with LC-Japanese are the 10 decimal digits in the range from U+0030 DIGIT ZERO to U+0039 DIGIT NINE.

In addition, the following characters from the Latin writing system are employable characters of LC-Japanese: the ranges from the U+0041 LATIN CAPITAL LETTER A character to the U+005A LATIN CAPITAL LETTER Z character (inclusive), and from the U+0061 LATIN SMALL LETTER A character to the U+007A LATIN SMALL LETTER Z character (inclusive).

In this version of FACR, LC-Japanese has a total of 6,597 employable characters.

For assistance in implementing a process that verifies whether a candidate string corresponding to a network name or a site name associated with a linguistic category complies with the employable character rules defined in this FACR specification, see Appendix C.1.

10.3.3. Arrangement Rules of LC-Japanese

The following rules are defined in accordance with the rules stated in Section 4.2.

The arrangement rules applicable to a network name associated with LC-Japanese are:

* The network name can contain only one type of connector character.

* If the type of connector character in the network name is the U+30FB KATAKANA MIDDLE DOT character, then each occurrence of that connector character is immediately preceded and followed by characters with the Hiragana, Katakana, or Han Unicode Script property.

  This arrangement rule is inspired by the rules concerning the use of characters defined in the policy document of the IDN table for the .jp ccTLD included in the Repository of Internationalized Domain Name (IDN) Practices maintained by the Internet Assigned Numbers Authority (IANA) [IANA-Repository].

* The network name contains at least one character with the Hiragana, Katakana, or Han Unicode Script property [UAX24].

  This arrangement rule is required by Section 6, since LC-Japanese includes characters borrowed from the writing system corresponding to LC-Latin.

  The characters used in the Kanji writing system are characters with the Han Unicode Script property.

The arrangement rules applicable to a site name associated with LC-Japanese are:

* If the network name contains a connector character, then the site name cannot contain a different type of connector character.

* If the network name does not contain a connector character, then the site name can contain only one type of connector character.

* The preceding arrangement rule applicable to the network name in a Frogans address concerning the U+30FB KATAKANA MIDDLE DOT character also applies to the site name.

The preceding arrangement rules concerning connector characters complement the rules concerning connector characters defined in the International Frogans Address Pattern (IFAP) specification [IFAP] (see IFAP, section 4.4. Connector Characters).

These arrangement rules of LC-Japanese for a site name have the same outcome irrespective of the preferred form of the network name, as required by Section 4.2.

For assistance in implementing a process that verifies whether a candidate string corresponding to a network name or a site name associated with a linguistic category complies with the arrangement rules defined in this FACR specification, see Appendix C.2.

10.3.4. Linguistic Categories Overlapping With LC-Japanese

In accordance with the rules stated in Section 6, LC-Japanese overlaps with another linguistic category if LC-Japanese has valid network names in common with that linguistic category. The validity of a network name associated with a linguistic category depends not only on the employable characters of that linguistic category, but also on its arrangement rules.

In order to determine whether LC-Japanese overlaps with other linguistic categories, the following employable characters and, when applicable, arrangement rules are taken into account:

A. The connector characters U+002D HYPHEN-MINUS and U+30FB KATAKANA MIDDLE DOT.

Some of these characters are also employable characters of other linguistic categories.

In accordance with Section 6, the mere fact that these characters are also employable characters of other linguistic categories does not cause LC-Japanese to overlap with these linguistic categories.

B. The 10 decimal digits in the range from U+0030 DIGIT ZERO to U+0039 DIGIT NINE.

These characters are also employable characters of other linguistic categories.

In accordance with Section 6, the mere fact that these characters are also employable characters of other linguistic categories does not cause LC-Japanese to overlap with these linguistic categories.

C. The characters borrowed from a writing system corresponding to another linguistic category.

These are characters borrowed from the writing system corresponding to LC-Latin. They are the 52 uppercase and lowercase alphabetical characters in the ranges from the U+0041 LATIN CAPITAL LETTER A character to the U+005A LATIN CAPITAL LETTER Z character (inclusive) and from the U+0061 LATIN SMALL LETTER A character to the U+007A LATIN SMALL LETTER Z character (inclusive).

These characters are also included in the employable characters of LC-Chinese, LC-Korean, and LC-Thai.

In accordance with Section 6, and given the arrangement rules of LC-Japanese, the mere fact that these characters are also employable characters of LC-Latin, LC-Chinese, LC-Korean, and LC-Thai does not cause LC-Chinese to overlap with these linguistic categories.

D. The characters from the writing systems corresponding to LC-Japanese that are borrowed by other linguistic categories.

With the exception of characters with the Han Unicode Script property, no employable characters of LC-Japanese that are also characters from one of the writing systems corresponding to LC-Japanese are borrowed by another linguistic category.

E. The characters with the Han Unicode Script property.

There are 6,358 of these characters, out of which:

- 6,181 are also employable characters of LC-Chinese
- 673 are also employable characters of LC-Korean

In accordance with Section 6, the mere fact that some of these characters are also employable characters of LC-Chinese and LC-Korean causes LC-Japanese to overlap with these linguistic categories.

As a result, LC-Japanese overlaps with the following linguistic categories: LC-Chinese and LC-Korean.

10.3.5. Intra-LC Convergence Form Types of LC-Japanese

The following rules are defined in accordance with the rules stated in Section 3.2 and Section 7.1.

There is only one type of Intra-LC convergence form that applies to network names and site names associated with LC-Japanese.

The Intra-LC convergence form type is defined using the Unicode Technical Standard #39 [UTS39] (see the Unicode Technical Standard #39, section 4 Confusable Detection) as a source, in accordance with Section 3.2.

The identifier of this Intra-LC convergence form type is the label 'Intra-LC-Japanese-Confusable'.

The Intra-LC convergence form of this type for a valid network name or a valid site name associated with LC-Japanese is the string of Unicode characters [Unicode] generated according to the rules stated in Section 7.1.

For assistance in implementing a function to generate Intra-LC convergence forms, see Appendix C.3.

10.4. LC-Korean

The label 'LC-Korean' is the identifier of the Korean linguistic category, which corresponds to a language that uses two writing systems.

This section defines, for LC-Korean, the corresponding language and writing systems, the employable characters, the arrangement rules, the overlapping linguistic categories, and the Intra-LC convergence form types.

10.4.1. Language and Writing Systems Corresponding to LC-Korean

The following rules are defined in accordance with the rules stated in Section 3.1.

The language corresponding to LC-Korean is the Korean language.

The writing systems corresponding to LC-Korean are:

* Hangul

* Hanja

Hanja corresponds to characters of Chinese origin.

10.4.2. Employable Characters of LC-Korean

In accordance with the rules stated in Section 4.1, the primary source used for determining the employable characters of LC-Korean is either an IDN table included in the Repository of Internationalized Domain Name (IDN) Practices maintained by the Internet Assigned Numbers Authority (IANA) [IANA-Repository], or data included in the Unicode Common Locale Data Repository (CLDR) maintained by the Unicode Consortium [CLDR].

In the Repository of IDN Practices, there is an IDN table for .kr [IDN-KR], the country-code Top-Level Domain (ccTLD) for the Republic of Korea. This IDN table is used for the registration of IDNs in the Korean language. It contains characters used in the Hangul writing system but it does not contain Hanja characters.

Therefore, for LC-Korean, the IDN table for the .kr ccTLD does not meet all the requirements in Section 4.1.

In accordance with the rules stated in Section 4.1, the primary source used for determining the employable characters of LC-Korean is data included in CLDR.

A character is an employable character of LC-Korean if all the following conditions are met:

1. The character is an eligible character according to the International Frogans Address Pattern specification (IFAP) [IFAP].

2. The character does not correspond to any of the following code points:

   * U+002A. This code point corresponds to the separator character in a Frogans address.

   * U+00B7. This code point corresponds to a connector character that can be used in the Hangul writing system, but is not included in the IDN table for the .kr ccTLD.

   * U+30FB. This code point corresponds to a connector character that belongs to another writing system.

3. The character is either accepted as a potential employable character by the method described in Section 4.1.2 where the script subtag is 'Kore' and the option to include auxiliary exemplar sets is enabled,

   or

   the character is in the ranges from the U+0041 LATIN CAPITAL LETTER A character to the U+005A LATIN CAPITAL LETTER Z character (inclusive) or from the U+0061 LATIN SMALL LETTER A character to the U+007A LATIN SMALL LETTER Z character (inclusive).

As a result, the only connector character that can be used in a network name or a site name associated with LC-Korean is the U+002D HYPHEN-MINUS character.

Furthermore, the decimal digits that can be used in a network name or a site name associated with LC-Korean are the 10 decimal digits in the range from U+0030 DIGIT ZERO to U+0039 DIGIT NINE.

In this version of FACR, LC-Korean has a total of 11235 employable characters.

For assistance in implementing a process that verifies whether a candidate string corresponding to a network name or a site name associated with a linguistic category complies with the employable character rules defined in this FACR specification, see Appendix C.1.

10.4.3. Arrangement Rules of LC-Korean

The following rules are defined in accordance with the rules stated in Section 4.2.

The arrangement rules applicable to a network name associated with LC-Korean are:

* The network name contains at least one character with the Hangul or Han Unicode Script property [UAX24].

  This arrangement rule is required by Section 6, since LC-Korean includes characters borrowed from the writing system corresponding to LC-Latin.

  The characters used in the Hanja writing system are characters with the Han Unicode Script property.

There are no arrangement rules applicable to a site name associated with LC-Korean.

For assistance in implementing a process that verifies whether a candidate string corresponding to a network name or a site name associated with a linguistic category complies with the arrangement rules defined in this FACR specification, see Appendix C.2.

10.4.4. Linguistic Categories Overlapping With LC-Korean

In accordance with the rules stated in Section 6, LC-Korean overlaps with another linguistic category if LC-Korean has valid network names in common with that linguistic category. The validity of a network name associated with a linguistic category depends not only on the employable characters of that linguistic category, but also on its arrangement rules.

In order to determine whether LC-Korean overlaps with other linguistic categories, the following employable characters and, when applicable, arrangement rules are taken into account:

A. The connector character U+002D HYPHEN-MINUS.

   This character is also an employable character of other linguistic categories.

   In accordance with Section 6, the mere fact that this character is also an employable character of other linguistic categories does not cause LC-Korean to overlap with these linguistic categories.

B. The 10 decimal digits in the range from U+0030 DIGIT ZERO to U+0039 DIGIT NINE.

These characters are also employable characters of other linguistic categories.

In accordance with Section 6, the mere fact that these characters are also employable characters of other linguistic categories does not cause LC-Korean to overlap with these linguistic categories.

C. The characters borrowed from a writing system corresponding to another linguistic category.

These are characters borrowed from the writing system corresponding to LC-Latin. They are the 52 uppercase and lowercase alphabetical characters in the ranges from the U+0041 LATIN CAPITAL LETTER A character to the U+005A LATIN CAPITAL LETTER Z character (inclusive) and from the U+0061 LATIN SMALL LETTER A character to the U+007A LATIN SMALL LETTER Z character (inclusive).

These characters are also included in the employable characters of LC-Chinese, LC-Japanese, and LC-Thai.

In accordance with Section 6, and given the arrangement rules of LC-Korean, the mere fact that these characters are also employable characters of LC-Latin, LC-Chinese, LC-Japanese, and LC-Thai does not cause LC-Korean to overlap with these linguistic categories.

D. The characters from the writing systems corresponding to LC-Korean that are borrowed by other linguistic categories.

With the exception of characters with the Han Unicode Script property, no employable characters of LC-Korean that are also characters from one of the writing systems corresponding to LC-Korean are borrowed by another linguistic category.

E. The characters with the Han Unicode Script property.

There are 755 of these characters, out of which:

- 752 are also employable characters of LC-Chinese
- 673 are also employable characters of LC-Japanese

In accordance with Section 6, the mere fact that some of these characters are also employable characters of LC-Chinese and

LC-Japanese causes LC-Korean to overlap with these linguistic categories.

As a result, LC-Korean overlaps with the following linguistic categories: LC-Chinese and LC-Japanese.

10.4.5. Intra-LC Convergence Form Types of LC-Korean

The following rules are defined in accordance with the rules stated in Section 3.2 and Section 7.1.

There is only one type of Intra-LC convergence form that applies to network names and site names associated with LC-Korean.

The Intra-LC convergence form type is defined using the Unicode Technical Standard #39 [UTS39] (see the Unicode Technical Standard #39, section 4 Confusable Detection) as a source, in accordance with Section 3.2.

The identifier of this Intra-LC convergence form type is the label 'Intra-LC-Korean-Confusable'.

The Intra-LC convergence form of this type for a valid network name or a valid site name associated with LC-Korean is the string of Unicode characters [Unicode] generated according to the rules stated in Section 7.1.

For assistance in implementing a function to generate Intra-LC convergence forms, see Appendix C.3.

10.5. LC-Arabic

The label 'LC-Arabic' is the identifier of the Arabic linguistic category, which corresponds to a group of languages that use the same writing system.

This section defines, for LC-Arabic, the corresponding languages and writing system, the employable characters, the arrangement rules, the overlapping linguistic categories, and the Intra-LC convergence form types.

10.5.1. Languages and Writing System Corresponding to LC-Arabic

The following rules are defined in accordance with the rules stated in Section 3.1.

The writing system corresponding to LC-Arabic is the Arabic writing system.

The languages corresponding to LC-Arabic are the languages listed in the Unicode Common Locale Data Repository (CLDR) [CLDR] for which the <language> element meets the following conditions:

* the value of the "scripts" attribute contains 'Arab', and

* the value of the "alt" attribute is not equal to 'secondary'

As a result, a total of 58 languages, including territorial variations, correspond to LC-Arabic. These languages comprise, among others, Arabic, Urdu, Punjabi, Persian, and Lahnda.

10.5.2. Employable Characters of LC-Arabic

In accordance with the rules stated in Section 4.1, the primary source used for determining the employable characters of LC-Arabic is either an IDN table included in the Repository of Internationalized Domain Name (IDN) Practices maintained by the Internet Assigned Numbers Authority (IANA) [IANA-Repository], or data included in the Unicode Common Locale Data Repository (CLDR) maintained by the Unicode Consortium [CLDR].

In the Repository of IDN Practices, there is no IDN table that contains all the characters commonly used in the languages corresponding to LC-Arabic.

Therefore, for LC-Arabic, no IDN table meets all the requirements in Section 4.1.

In accordance with the rules stated in Section 4.1, the primary source used for determining the employable characters of LC-Arabic is data included in CLDR.

A character is an employable character of LC-Arabic if all the following conditions are met:

1. The character is an eligible character according to the International Frogans Address Pattern specification (IFAP) [IFAP].

2. The character does not correspond to any of the following code points:

   * U+002A. This code point corresponds to the separator character in a Frogans address.

   * Code points in the range from U+064B to U+0652 (inclusive). These code points correspond to Tashkeel and Shadda accent marks. They are excluded as proposed in RFC 5564 [RFC5564].

   * U+0654, U+0655, U+0670, U+0656, U+0657, U+065A, U+065B, U+06EA, U+06ED. These code points correspond to combining marks.

3. The character is accepted as a potential employable character by the method described in Section 4.1.2 where the script subtag is 'Arab' and the option to include auxiliary exemplar sets is disabled.

As a result, the only connector character that can be used in a network name or a site name associated with LC-Arabic is the U+002D HYPHEN-MINUS character.

Furthermore, the decimal digits that can be used in a network name or a site name associated with LC-Arabic are the 30 decimal digits in the ranges from U+0030 DIGIT ZERO to U+0039 DIGIT NINE, from U+0660 ARABIC-INDIC DIGIT ZERO to U+0669 ARABIC-INDIC DIGIT NINE, and from U+06F0 EXTENDED ARABIC-INDIC DIGIT ZERO to U+06F9 EXTENDED ARABIC-INDIC DIGIT NINE.

In this version of FACR, LC-Arabic has a total of 102 employable characters. According to the data in Unicode CLDR, these employable characters can be used with at least 8 of the 58 languages corresponding to LC-Arabic.

For assistance in implementing a process that verifies whether a candidate string corresponding to a network name or a site name associated with a linguistic category complies with the employable character rules defined in this FACR specification, see Appendix C.1.

10.5.3. Arrangement Rules of LC-Arabic

The following rules are defined in accordance with the rules stated in Section 4.2.

The arrangement rules applicable to a network name associated with LC-Arabic are:

* If the network name contains decimal digits, then all these decimal digits belong to the same numbering system.

  This arrangement rule is inspired by the rules concerning the use of numerals defined in RFC 5564 [RFC5564] (see RFC 5564, section 2.3.1 Numerals).

The arrangement rules applicable to a site name associated with LC-Arabic are:

* If the network name contains a decimal digit, then the site name cannot contain a decimal digit that belongs to a different numbering system.

* If the network name does not contain a decimal digit and the site name contains decimal digits, then all these decimal digits belong to the same numbering system.

These arrangement rules of LC-Arabic for a site name have the same outcome irrespective of the preferred form of the network name, as required by Section 4.2.

For assistance in implementing a process that verifies whether a candidate string corresponding to a network name or a site name associated with a linguistic category complies with the arrangement rules defined in this FACR specification, see Appendix C.2.

10.5.4. Linguistic Categories Overlapping With LC-Arabic

In accordance with the rules stated in Section 6, LC-Arabic overlaps with another linguistic category if LC-Arabic has valid network names in common with that linguistic category. The validity of a network name associated with a linguistic category depends not only on the employable characters of that linguistic category, but also on its arrangement rules.

In order to determine whether LC-Arabic overlaps with other linguistic categories, the following employable characters and, when applicable, arrangement rules are taken into account:

A. The connector character U+002D HYPHEN-MINUS.

This character is also an employable character of other linguistic categories.

In accordance with Section 6, the mere fact that this character is also an employable character of other linguistic categories does not cause LC-Arabic to overlap with these linguistic categories.

B. The 30 decimal digits in the ranges from U+0030 DIGIT ZERO to U+0039 DIGIT NINE, from U+0660 ARABIC-INDIC ZERO to U+0669 ARABIC-INDIC NINE, and from U+06F0 EXTENDED ARABIC-INDIC ZERO to U+06F9 EXTENDED ARABIC-INDIC NINE.

Some of these characters are also employable characters of other linguistic categories.

In accordance with Section 6, the mere fact that some of these characters are also employable characters of other linguistic categories does not cause LC-Arabic to overlap with these linguistic categories.

C. The characters borrowed from a writing system corresponding to another linguistic category.

No employable characters of LC-Arabic are characters borrowed from a writing system corresponding to another linguistic category.

D. The characters from the writing system corresponding to LC-Arabic that are borrowed by other linguistic categories.

No employable characters of LC-Arabic that are also characters from the writing system corresponding to LC-Arabic are borrowed by another linguistic category.

E. The characters with the Han Unicode Script property [UAX24].

No employable characters of LC-Arabic are characters with the Han Unicode Script property.

As a result, LC-Arabic does not overlap with any other linguistic category.

10.5.5. Intra-LC Convergence Form Types of LC-Arabic

The following rules are defined in accordance with the rules stated in Section 3.2 and Section 7.1.

There is only one type of Intra-LC convergence form that applies to network names and site names associated with LC-Arabic.

The Intra-LC convergence form type is defined using the Unicode Technical Standard #39 [UTS39] (see the Unicode Technical Standard #39, section 4 Confusable Detection) as a source, in accordance with Section 3.2.

The identifier of this Intra-LC convergence form type is the label 'Intra-LC-Arabic-Confusable'.

The Intra-LC convergence form of this type for a valid network name or a valid site name associated with LC-Arabic is the string of Unicode characters [Unicode] generated according to the rules stated in Section 7.1.

For assistance in implementing a function to generate Intra-LC convergence forms, see Appendix C.3.

10.6. LC-Cyrillic

The label 'LC-Cyrillic' is the identifier of the Cyrillic linguistic category, which corresponds to a group of languages that use the same writing system.

This section defines, for LC-Cyrillic, the corresponding languages and writing system, the employable characters, the arrangement rules, the overlapping linguistic categories, and the Intra-LC convergence form types.

10.6.1. Languages and Writing System Corresponding to LC-Cyrillic

The following rules are defined in accordance with the rules stated in Section 3.1.

The writing system corresponding to LC-Cyrillic is the Cyrillic writing system.

The languages corresponding to LC-Cyrillic are the languages listed in the Unicode Common Locale Data Repository (CLDR) [CLDR] for which the <language> element meets the following conditions:

* the value of the "scripts" attribute contains 'Cyrl', and

* the value of the "alt" attribute is not equal to 'secondary'

As a result, a total of 65 languages, including territorial variations, correspond to LC-Cyrillic. These languages comprise, among others, Russian, Ukrainian, Kazakh and Belarusian.

10.6.2. Employable Characters of LC-Cyrillic

In accordance with the rules stated in Section 4.1, the primary source used for determining the employable characters of LC-Cyrillic is either an IDN table included in the Repository of Internationalized Domain Name (IDN) Practices maintained by the Internet Assigned Numbers Authority (IANA) [IANA-Repository], or data included in the Unicode Common Locale Data Repository (CLDR) maintained by the Unicode Consortium [CLDR].

In the Repository of IDN Practices, there is no IDN table that contains all the characters commonly used in the languages corresponding to LC-Cyrillic.

Therefore, for LC-Cyrillic, no IDN table meets all the requirements in Section 4.1.

In accordance with the rules stated in Section 4.1, the primary source used for determining the employable characters of LC-Cyrillic is data included in CLDR.

A character is an employable character of LC-Cyrillic if all the following conditions are met:

1. The character is an eligible character according to the International Frogans Address Pattern specification (IFAP) [IFAP].

2. The character does not correspond to the following code point: U+002A. This code point corresponds to the separator character in a Frogans address.

3. The character is accepted as a potential employable character by the method described in Section 4.1.2 where the script subtag is 'Cyrl' and the option to include auxiliary exemplar sets is disabled.

As a result, the only connector character that can be used in a network name or a site name associated with LC-Cyrillic is the U+002D HYPHEN-MINUS character.

Furthermore, the decimal digits that can be used in a network name or a site name associated with LC-Cyrillic are the 10 decimal digits in the range from U+0030 DIGIT ZERO to U+0039 DIGIT NINE.

In this version of FACR, LC-Cyrillic has a total of 136 employable characters. According to the data in Unicode CLDR, these employable characters can be used with at least 14 of the 65 languages corresponding to LC-Cyrillic.

For assistance in implementing a process that verifies whether a candidate string corresponding to a network name or a site name associated with a linguistic category complies with the employable character rules defined in this FACR specification, see Appendix C.1.

10.6.3. Arrangement Rules of LC-Cyrillic

The following rules are defined in accordance with the rules stated in Section 4.2.

There are no arrangement rules applicable to a network name associated with LC-Cyrillic.

There are no arrangement rules applicable to a site name associated with LC-Cyrillic.

For assistance in implementing a process that verifies whether a candidate string corresponding to a network name or a site name associated with a linguistic category complies with the arrangement rules defined in this FACR specification, see Appendix C.2.

10.6.4. Linguistic Categories Overlapping With LC-Cyrillic

In accordance with the rules stated in Section 6, LC-Cyrillic overlaps with another linguistic category if LC-Cyrillic has valid network names in common with that linguistic category. The validity of a network name associated with a linguistic category depends not only on the employable characters of that linguistic category, but also on its arrangement rules.

In order to determine whether LC-Cyrillic overlaps with other linguistic categories, the following employable characters and, when applicable, arrangement rules are taken into account:

A. The connector character U+002D HYPHEN-MINUS.

This character is also an employable character of other linguistic categories.

In accordance with Section 6, the mere fact that this character is also an employable character of other linguistic categories does not cause LC-Cyrillic to overlap with these linguistic categories.

B. The 10 decimal digits in the range from U+0030 DIGIT ZERO to U+0039 DIGIT NINE.

These characters are also employable characters of other linguistic categories.

In accordance with Section 6, the mere fact that these characters are also employable characters of other linguistic categories does not cause LC-Cyrillic to overlap with these linguistic categories.

C. The characters borrowed from a writing system corresponding to another linguistic category.

No employable characters of LC-Cyrillic are characters borrowed from a writing system corresponding to another linguistic category.

D. The characters from the writing system corresponding to LC-Cyrillic that are borrowed by other linguistic categories.

No employable characters of LC-Cyrillic that are also characters from the writing system corresponding to LC-Cyrillic are borrowed by another linguistic category.

E. The characters with the Han Unicode Script property [UAX24].

No employable characters of LC-Cyrillic are characters with the Han Unicode Script property.

As a result, LC-Cyrillic does not overlap with any other linguistic category.

10.6.5. Intra-LC Convergence Form Types of LC-Cyrillic

The following rules are defined in accordance with the rules stated in Section 3.2 and Section 7.1.

There is only one type of Intra-LC convergence form that applies to network names and site names associated with LC-Cyrillic.

The Intra-LC convergence form type is defined using the Unicode Technical Standard #39 [UTS39] (see the Unicode Technical Standard #39, section 4 Confusable Detection) as a source, in accordance with Section 3.2.

The identifier of this Intra-LC convergence form type is the label 'Intra-LC-Cyrillic-Confusable'.

The Intra-LC convergence form of this type for a valid network name or a valid site name associated with LC-Cyrillic is the string of Unicode characters [Unicode] generated according to the rules stated in Section 7.1.

For assistance in implementing a function to generate Intra-LC convergence forms, see Appendix C.3.

10.7. LC-Hebrew

The label 'LC-Hebrew' is the identifier of the Hebrew linguistic category, which corresponds to a group of languages that use the same writing system.

This section defines, for LC-Hebrew, the corresponding languages and writing system, the employable characters, the arrangement rules, the overlapping linguistic categories, and the Intra-LC convergence form types.

10.7.1. Languages and Writing System Corresponding to LC-Hebrew

The following rules are defined in accordance with the rules stated in Section 3.1.

The writing system corresponding to LC-Hebrew is the Hebrew writing system.

The languages corresponding to LC-Hebrew are the languages listed in the Unicode Common Locale Data Repository (CLDR) [CLDR] for which the <language> element meets the following conditions:

* the value of the "scripts" attribute contains 'Hebr', and

* the value of the "alt" attribute is not equal to 'secondary'

As a result, a total of five languages, including territorial variations, correspond to LC-Hebrew. These languages comprise Hebrew, Yiddish, Ladino, Judeo-Persian, and Judeo-Arabic.

10.7.2. Employable Characters of LC-Hebrew

In accordance with the rules stated in Section 4.1, the primary source used for determining the employable characters of LC-Hebrew is either an IDN table included in the Repository of Internationalized Domain Name (IDN) Practices maintained by the Internet Assigned Numbers Authority (IANA) [IANA-Repository], or data included in the Unicode Common Locale Data Repository (CLDR) maintained by the Unicode Consortium [CLDR].

In the Repository of IDN Practices, there is an IDN table for .il [IDN-HE], the country-code Top-Level Domain (ccTLD) for Israel. This IDN table is used for the registration of IDNs in the Hebrew language. It contains the characters commonly used in the Hebrew language.

Therefore, for LC-Hebrew, it is considered that the IDN table for the

.il ccTLD meets all the requirements in Section 4.1.

In accordance with the rules stated in Section 4.1, the primary source used for determining the employable characters of LC-Hebrew is the IDN table for the .il ccTLD.

A character is an employable character of LC-Hebrew if all the following conditions are met:

1. The character is an eligible character according to the International Frogans Address Pattern specification (IFAP) [IFAP].

2. The character does not correspond to the code point U+002A which is the separator character in a Frogans address.

3. The character is accepted as a potential employable character by the method described in Section 4.1.1 where the IDN table is the IDN table for the .il ccTLD.

As a result, the only connector character that can be used in a network name or a site name associated with LC-Hebrew is the U+002D HYPHEN-MINUS character.

Furthermore, the decimal digits that can be used in a network name or a site name associated with LC-Hebrew are the 10 decimal digits in the range from U+0030 DIGIT ZERO to U+0039 DIGIT NINE.

In this version of FACR, LC-Hebrew has a total of 38 employable characters. According to the data in Unicode CLDR, these employable characters can be used with at least two of the five languages corresponding to LC-Hebrew.

For assistance in implementing a process that verifies whether a candidate string corresponding to a network name or a site name associated with a linguistic category complies with the employable character rules defined in this FACR specification, see Appendix C.1.

10.7.3. Arrangement Rules of LC-Hebrew

The following rules are defined in accordance with the rules stated in Section 4.2.

There are no arrangement rules applicable to a network name associated with LC-Hebrew.

There are no arrangement rules applicable to a site name associated with LC-Hebrew.

For assistance in implementing a process that verifies whether a candidate string corresponding to a network name or a site name associated with a linguistic category complies with the arrangement rules defined in this FACR specification, see Appendix C.2.

10.7.4. Linguistic Categories Overlapping With LC-Hebrew

In accordance with the rules stated in Section 6, LC-Hebrew overlaps with another linguistic category if LC-Hebrew has valid network names in common with that linguistic category. The validity of a network name associated with a linguistic category depends not only on the employable characters of that linguistic category, but also on its arrangement rules.

In order to determine whether LC-Hebrew overlaps with other linguistic categories, the following employable characters and, when applicable, arrangement rules are taken into account:

A. The connector character U+002D HYPHEN-MINUS.

This character is also an employable character of other linguistic categories.

In accordance with Section 6, the mere fact that this character is also an employable character of other linguistic categories does not cause LC-Hebrew to overlap with these linguistic categories.

B. The 10 decimal digits in the range from U+0030 DIGIT ZERO to U+0039 DIGIT NINE.

These characters are also employable characters of other linguistic categories.

In accordance with Section 6, the mere fact that these characters are also employable characters of other linguistic categories does not cause LC-Hebrew to overlap with these linguistic categories.

C. The characters borrowed from a writing system corresponding to another linguistic category.

No employable characters of LC-Hebrew are characters borrowed from a writing system corresponding to another linguistic category.

D. The characters from the writing system corresponding to LC-Hebrew that are borrowed by other linguistic categories.

No employable characters of LC-Hebrew that are also characters from the writing system corresponding to LC-Hebrew are borrowed by another linguistic category.

E. The characters with the Han Unicode Script property [UAX24].

No employable characters of LC-Hebrew are characters with the Han Unicode Script property.

As a result, LC-Hebrew does not overlap with any other linguistic category.

10.7.5. Intra-LC Convergence Form Types of LC-Hebrew

The following rules are defined in accordance with the rules stated in Section 3.2 and Section 7.1.

There is only one type of Intra-LC convergence form that applies to network names and site names associated with LC-Hebrew.

The Intra-LC convergence form type is defined using the Unicode Technical Standard #39 [UTS39] (see the Unicode Technical Standard #39, section 4 Confusable Detection) as a source, in accordance with Section 3.2.

The identifier of this Intra-LC convergence form type is the label 'Intra-LC-Hebrew-Confusable'.

The Intra-LC convergence form of this type for a valid network name or a valid site name associated with LC-Hebrew is the string of Unicode characters [Unicode] generated according to the rules stated in Section 7.1.

For assistance in implementing a function to generate Intra-LC convergence forms, see Appendix C.3.

10.8. LC-Devanagari

The label 'LC-Devanagari' is the identifier of the Devanagari linguistic category, which corresponds to a group of languages that use the same writing system.

This section defines, for LC-Devanagari, the corresponding languages and writing system, the employable characters, the arrangement rules, the overlapping linguistic categories, and the Intra-LC convergence form types.

10.8.1. Languages and Writing System Corresponding to LC-Devanagari

The following rules are defined in accordance with the rules stated in Section 3.1.

The writing system corresponding to LC-Devanagari is the Devanagari writing system.

The languages corresponding to LC-Devanagari are the languages listed in the Unicode Common Locale Data Repository (CLDR) [CLDR] for which the <language> element meets the following conditions:

* the value of the "scripts" attribute contains 'Deva', and

* the value of the "alt" attribute is not equal to 'secondary'

As a result, a total of 59 languages, including territorial variations, correspond to LC-Devanagari. These languages comprise, among others, Hindi, Marathi, Bhojpuri, Awadhi, and Nepali.

10.8.2. Employable Characters of LC-Devanagari

In accordance with the rules stated in Section 4.1, the primary source used for determining the employable characters of LC-Devanagari is either an IDN table included in the Repository of Internationalized Domain Name (IDN) Practices maintained by the Internet Assigned Numbers Authority (IANA) [IANA-Repository], or data included in the Unicode Common Locale Data Repository (CLDR) maintained by the Unicode Consortium [CLDR].

In the Repository of IDN Practices, there is no IDN table that contains all the characters commonly used in the languages corresponding to LC-Devanagari.

Therefore, for LC-Devanagari, no IDN table meets all the requirements in Section 4.1.

In accordance with the rules stated in Section 4.1, the primary source used for determining the employable characters of LC-Devanagari is data included in CLDR.

A character is an employable character of LC-Devanagari if all the following conditions are met:

1. The character is an eligible character according to the International Frogans Address Pattern specification (IFAP) [IFAP].

2. The character does not correspond to the following code point: U+002A. This code point corresponds to the separator character in a Frogans address.

3. The character is accepted as a potential employable character by the method described in Section 4.1.2 where the script subtag is 'Deva' and the option to include auxiliary exemplar sets is disabled.

As a result, the only connector character that can be used in a network name or a site name associated with LC-Devanagari is the U+002D HYPHEN-MINUS character.

Furthermore, the decimal digits that can be used in a network name or a site name associated with LC-Devanagari are the 20 decimal digits in the ranges from U+0030 DIGIT ZERO to U+0039 DIGIT NINE and from U+0966 DEVANAGARI DIGIT ZERO to U+096F DEVANAGARI DIGIT NINE.

In this version of FACR, LC-Devanagari has a total of 89 employable characters. According to the data in Unicode CLDR, these employable characters can be used with at least 5 of the 59 languages corresponding to LC-Devanagari.

For assistance in implementing a process that verifies whether a candidate string corresponding to a network name or a site name associated with a linguistic category complies with the employable character rules defined in this FACR specification, see Appendix C.1.

10.8.3. Arrangement Rules of LC-Devanagari

The following rules are defined in accordance with the rules stated in Section 4.2.

The arrangement rules applicable to a network name associated with LC-Devanagari are:

* If the network name contains decimal digits, then all these decimal digits belong to the same numbering system.

  This arrangement rule is inspired by the rule for the use of different decimal number systems defined in the Unicode Technical Standard #39 [UTS39] (see the Unicode Technical Standard #39, section 5.3 Mixed-Number Detection).

The arrangement rules applicable to a site name associated with LC-Devanagari are:

* If the network name contains a decimal digit, then the site name cannot contain a decimal digit that belongs to a different numbering system.

* If the network name does not contain a decimal digit and the site name contains decimal digits, then all these decimal digits belong to the same numbering system.

These arrangement rules of LC-Devanagari for a site name have the same outcome irrespective of the preferred form of the network name, as required by Section 4.2.

For assistance in implementing a process that verifies whether a candidate string corresponding to a network name or a site name associated with a linguistic category complies with the arrangement rules defined in this FACR specification, see Appendix C.2.

10.8.4. Linguistic Categories Overlapping With LC-Devanagari

In accordance with the rules stated in Section 6, LC-Devanagari overlaps with another linguistic category if LC-Devanagari has valid network names in common with that linguistic category. The validity of a network name associated with a linguistic category depends not only on the employable characters of that linguistic category, but also on its arrangement rules.

In order to determine whether LC-Devanagari overlaps with other linguistic categories, the following employable characters and, when applicable, arrangement rules are taken into account:

A. The connector character U+002D HYPHEN-MINUS.

   This character is also an employable character of other linguistic categories.

   In accordance with Section 6, the mere fact that this character is also an employable character of other linguistic categories does not cause LC-Devanagari to overlap with these linguistic categories.

B. The 20 decimal digits in the ranges from U+0030 DIGIT ZERO to U+0039 DIGIT NINE and from U+0966 DEVANAGARI DIGIT ZERO to U+096F DEVANAGARI DIGIT NINE.

These characters are also employable characters of other linguistic categories.

In accordance with Section 6, the mere fact that these characters are also employable characters of other linguistic categories does not cause LC-Devanagari to overlap with these linguistic categories.

C. The characters borrowed from a writing system corresponding to another linguistic category.

No employable characters of LC-Devanagari are characters borrowed from a writing system corresponding to another linguistic category.

D. The characters from the writing system corresponding to LC-Devanagari that are borrowed by other linguistic categories.

No employable characters of LC-Devanagari that are also characters from the writing system corresponding to LC-Devanagari are borrowed by another linguistic category.

E. The characters with the Han Unicode Script property [UAX24].

No employable characters of LC-Devanagari are characters with the Han Unicode Script property.

As a result, LC-Devanagari does not overlap with any other linguistic category.

10.8.5. Intra-LC Convergence Form Types of LC-Devanagari

The following rules are defined in accordance with the rules stated in Section 3.2 and Section 7.1.

There is only one type of Intra-LC convergence form that applies to network names and site names associated with LC-Devanagari.

The Intra-LC convergence form type is defined using the Unicode Technical Standard #39 [UTS39] (see the Unicode Technical Standard #39, section 4 Confusable Detection) as a source, in accordance with

Section 3.2.

The identifier of this Intra-LC convergence form type is the label 'Intra-LC-Devanagari-Confusable'.

The Intra-LC convergence form of this type for a valid network name or a valid site name associated with LC-Devanagari is the string of Unicode characters [Unicode] generated according to the rules stated in Section 7.1.

For assistance in implementing a function to generate Intra-LC convergence forms, see Appendix C.3.

10.9. LC-Thai

The label 'LC-Thai' is the identifier of the Thai linguistic category, which corresponds to a group of languages that use the same writing system.

This section defines, for LC-Thai, the corresponding languages and writing system, the employable characters, the arrangement rules, the overlapping linguistic categories, and the Intra-LC convergence form types.

10.9.1. Languages and Writing System Corresponding to LC-Thai

The following rules are defined in accordance with the rules stated in Section 3.1.

The writing system corresponding to LC-Thai is the Thai writing system.

The languages corresponding to LC-Thai are the languages listed in the Unicode Common Locale Data Repository (CLDR) [CLDR] for which the <language> element meets the following conditions:

* the value of the "scripts" attribute contains 'Thai', and

* the value of the "alt" attribute is not equal to 'secondary'

As a result, a total of seven languages, including territorial variations, correspond to LC-Thai. These languages comprise Thai, Northeastern Thai, Southern Thai, Northern Khmer, Kuy, Western Lawa, and Eastern Lawa.

10.9.2. Employable Characters of LC-Thai

In accordance with the rules stated in Section 4.1, the primary source used for determining the employable characters of LC-Thai is either an IDN table included in the Repository of Internationalized Domain Name (IDN) Practices maintained by the Internet Assigned Numbers Authority (IANA) [IANA-Repository], or data included in the Unicode Common Locale Data Repository (CLDR) maintained by the Unicode Consortium [CLDR].

In the Repository of IDN Practices, there is an IDN table for .th [IDN-TH], the country-code Top-Level Domain (ccTLD) for the Kingdom of Thailand. This IDN table is used for the registration of IDNs in the Thai language. It contains characters used in the Thai writing system.

Therefore, for LC-Thai, it is considered that the IDN table for the .th ccTLD meets all the requirements in Section 4.1.

In accordance with the rules stated in Section 4.1, the primary source used for determining the employable characters of LC-Thai is the IDN table for the .th ccTLD.

A character is an employable character of LC-Thai if all the following conditions are met:

1. The character is an eligible character according to the International Frogans Address Pattern specification (IFAP) [IFAP].

2. The character does not correspond to the following code point: U+002A. This code point corresponds to the separator character in a Frogans address.

3. The character is accepted as a potential employable character by the method described in Section 4.1.1 where the IDN table is the IDN table for the .th ccTLD.

As a result, the only connector character that can be used in a network name or a site name associated with LC-Thai is the U+002D HYPHEN-MINUS character.

Furthermore, the decimal digits that can be used in a network name or a site name associated with LC-Thai are the 20 decimal digits in the ranges from U+0030 DIGIT ZERO to U+0039 DIGIT NINE and from U+0E50 THAI DIGIT ZERO to U+0E59 THAI DIGIT NINE.

In addition, the following characters from the Latin writing system are employable characters of LC-Thai: the ranges from the U+0041 LATIN CAPITAL LETTER A character to the U+005A LATIN CAPITAL LETTER Z character (inclusive), and from the U+0061 LATIN SMALL LETTER A character to the U+007A LATIN SMALL LETTER Z character (inclusive).

In this version of FACR, LC-Thai has a total of 144 employable characters. According to the data in Unicode CLDR, these employable characters can be used with at least one of the seven languages corresponding to LC-Thai.

For assistance in implementing a process that verifies whether a candidate string corresponding to a network name or a site name associated with a linguistic category complies with the employable character rules defined in this FACR specification, see Appendix C.1.

10.9.3. Arrangement Rules of LC-Thai

The following rules are defined in accordance with the rules stated in Section 4.2.

The arrangement rules applicable to a network name associated with LC-Thai are:

* If the network name contains decimal digits, then all these decimal digits belong to the same numbering system.

  This arrangement rule is inspired by the rule for the use of different decimal number systems defined in the Unicode Technical Standard #39 [UTS39] (see the Unicode Technical Standard #39, section 5.3 Mixed-Number Detection).

* The network name contains at least one character with the Thai Unicode Script property [UAX24].

  This arrangement rule is required by Section 6, since LC-Thai includes characters borrowed from the writing system corresponding to LC-Latin.

The arrangement rules applicable to a site name associated with LC-Thai are:

* If the network name contains a decimal digit, then the site name cannot contain a decimal digit that belongs to a different numbering system.

* If the network name does not contain a decimal digit and the site name contains decimal digits, then all these decimal digits belong to the same numbering system.

These arrangement rules of LC-Thai for a site name have the same outcome irrespective of the preferred form of the network name, as required by Section 4.2.

For assistance in implementing a process that verifies whether a candidate string corresponding to a network name or a site name associated with a linguistic category complies with the arrangement rules defined in this FACR specification, see Appendix C.2.

10.9.4. Linguistic Categories Overlapping With LC-Thai

In accordance with the rules stated in Section 6, LC-Thai overlaps with another linguistic category if LC-Thai has valid network names in common with that linguistic category. The validity of a network name associated with a linguistic category depends not only on the employable characters of that linguistic category, but also on its arrangement rules.

In order to determine whether LC-Thai overlaps with other linguistic categories, the following employable characters and, when applicable, arrangement rules are taken into account:

A. The connector character U+002D HYPHEN-MINUS.

   This character is also an employable character of other linguistic categories.

   In accordance with Section 6, the mere fact that this character is also an employable character of other linguistic categories does not cause LC-Thai to overlap with these linguistic categories.

B. The 20 decimal digits in the ranges from U+0030 DIGIT ZERO to U+0039 DIGIT NINE and from U+0E50 THAI DIGIT ZERO to U+0E59 THAI DIGIT NINE.

   These characters are also employable characters of other linguistic categories.

   In accordance with Section 6, the mere fact that these characters are also employable characters of other linguistic categories does not cause LC-Thai to overlap with these linguistic categories.

C. The characters borrowed from a writing system corresponding to another linguistic category.

   These are characters borrowed from the writing system corresponding to LC-Latin. They are the 52 uppercase and lowercase alphabetical characters in the ranges from the U+0041 LATIN CAPITAL LETTER A character to the U+005A LATIN CAPITAL LETTER Z character (inclusive) and from the U+0061 LATIN SMALL LETTER A character to the U+007A LATIN SMALL LETTER Z character (inclusive).

   These characters are also included in the employable characters of LC-Chinese, LC-Japanese, and LC-Korean.

   In accordance with Section 6, and given the arrangement rules of LC-Thai, the mere fact that these characters are also employable characters of LC-Latin, LC-Chinese, LC-Japanese, and LC-Korean does not cause LC-Thai to overlap with these linguistic categories.

D. The characters from the writing system corresponding to LC-Thai that are borrowed by other linguistic categories.

No employable characters of LC-Thai are characters borrowed from a writing system corresponding to another linguistic category.

E. The characters with the Han Unicode Script property.

No employable characters of LC-Thai are characters with the Han Unicode Script property.

As a result, LC-Thai does not overlap with any other linguistic category.

10.9.5. Intra-LC Convergence Form Types of LC-Thai

The following rules are defined in accordance with the rules stated in Section 3.2 and Section 7.1.

There is only one type of Intra-LC convergence form that applies to network names and site names associated with LC-Thai.

The Intra-LC convergence form type is defined using the Unicode Technical Standard #39 [UTS39] (see the Unicode Technical Standard #39, section 4 Confusable Detection) as a source, in accordance with Section 3.2.

The identifier of this Intra-LC convergence form type is the label 'Intra-LC-Thai-Confusable'.

The Intra-LC convergence form of this type for a valid network name or a valid site name associated with LC-Thai is the string of Unicode characters [Unicode] generated according to the rules stated in Section 7.1.

For assistance in implementing a function to generate Intra-LC convergence forms, see Appendix C.3.

10.10. LC-Greek

The label 'LC-Greek' is the identifier of the Greek linguistic category, which corresponds to a group of languages that use the same writing system.

This section defines, for LC-Greek, the corresponding languages and writing system, the employable characters, the arrangement rules, the overlapping linguistic categories, and the Intra-LC convergence form types.

10.10.1. Languages and Writing System Corresponding to LC-Greek

The following rules are defined in accordance with the rules stated in Section 3.1.

The writing system corresponding to LC-Greek is the Greek writing system.

The languages corresponding to LC-Greek are the languages listed in the Unicode Common Locale Data Repository (CLDR) [CLDR] for which the <language> element meets the following conditions:

* the value of the "scripts" attribute contains 'Grek', and

* the value of the "alt" attribute is not equal to 'secondary'

As a result, a total of four languages, including territorial variations, correspond to LC-Greek. These languages comprise Modern Greek, Pontic Greek, Balkan Gagauz Turkish, and Tsakonian.

10.10.2. Employable Characters of LC-Greek

In accordance with the rules stated in Section 4.1, the primary source used for determining the employable characters of LC-Greek is either an IDN table included in the Repository of Internationalized Domain Name (IDN) Practices maintained by the Internet Assigned Numbers Authority (IANA) [IANA-Repository], or data included in the Unicode Common Locale Data Repository (CLDR) maintained by the Unicode Consortium [CLDR].

In the Repository of IDN Practices, there is no IDN table that contains all the characters commonly used in the languages corresponding to LC-Greek.

Therefore, for LC-Greek, no IDN table meets all the requirements in Section 4.1.

In accordance with the rules stated in Section 4.1, the primary source used for determining the employable characters of LC-Greek is data included in CLDR.

A character is an employable character of LC-Greek if all the following conditions are met:

1. The character is an eligible character according to the International Frogans Address Pattern specification (IFAP) [IFAP].

2. The character does not correspond to the following code point: U+002A. This code point corresponds to the separator character in a Frogans address.

3. The character is accepted as a potential employable character by the method described in Section 4.1.2 where the script subtag is 'Grek' and the option to include auxiliary exemplar sets is disabled.

As a result, the only connector character that can be used in a network name or a site name associated with LC-Greek is the U+002D HYPHEN-MINUS character.

Furthermore, the decimal digits that can be used in a network name or a site name associated with LC-Greek are the 10 decimal digits in the range from U+0030 DIGIT ZERO to U+0039 DIGIT NINE.

In this version of FACR, LC-Greek has a total of 82 employable characters. According to the data in Unicode CLDR, these employable characters can be used with at least one of the four languages corresponding to LC-Greek.

For assistance in implementing a process that verifies whether a candidate string corresponding to a network name or a site name associated with a linguistic category complies with the employable character rules defined in this FACR specification, see Appendix C.1.

10.10.3. Arrangement Rules of LC-Greek

The following rules are defined in accordance with the rules stated in Section 4.2.

There are no arrangement rules applicable to a network name associated with LC-Greek.

There are no arrangement rules applicable to a site name associated with LC-Greek.

For assistance in implementing a process that verifies whether a candidate string corresponding to a network name or a site name associated with a linguistic category complies with the arrangement rules defined in this FACR specification, see Appendix C.2.

10.10.4. Linguistic Categories Overlapping With LC-Greek

In accordance with the rules stated in Section 6, LC-Greek overlaps with another linguistic category if LC-Greek has valid network names in common with that linguistic category. The validity of a network name associated with a linguistic category depends not only on the employable characters of that linguistic category, but also on its arrangement rules.

In order to determine whether LC-Greek overlaps with other linguistic categories, the following employable characters and, when applicable, arrangement rules are taken into account:

A. The connector character U+002D HYPHEN-MINUS.

This character is also an employable character of other linguistic categories.

In accordance with Section 6, the mere fact that this character is also an employable character of other linguistic categories does not cause LC-Greek to overlap with these linguistic categories.

B. The 10 decimal digits in the range from U+0030 DIGIT ZERO to U+0039 DIGIT NINE.

These characters are also employable characters of other linguistic categories.

In accordance with Section 6, the mere fact that these characters are also employable characters of other linguistic categories does not cause LC-Greek to overlap with these linguistic categories.

C. The characters borrowed from a writing system corresponding to another linguistic category.

No employable characters of LC-Greek are characters borrowed from a writing system corresponding to another linguistic category.

D. The characters from the writing system corresponding to LC-Greek that are borrowed by other linguistic categories.

No employable characters of LC-Greek that are also characters from the writing system corresponding to LC-Greek are borrowed by another linguistic category.

E. The characters with the Han Unicode Script property [UAX24].

No employable characters of LC-Greek are characters with the Han Unicode Script property.

As a result, LC-Greek does not overlap with any other linguistic category.

10.10.5. Intra-LC Convergence Form Types of LC-Greek

The following rules are defined in accordance with the rules stated in Section 3.2 and Section 7.1.

There is only one type of Intra-LC convergence form that applies to network names and site names associated with LC-Greek.

The Intra-LC convergence form type is defined using the Unicode Technical Standard #39 [UTS39] (see the Unicode Technical Standard #39, section 4 Confusable Detection) as a source, in accordance with Section 3.2.

The identifier of this Intra-LC convergence form type is the label 'Intra-LC-Greek-Confusable'.

The Intra-LC convergence form of this type for a valid network name or a valid site name associated with LC-Greek is the string of Unicode characters [Unicode] generated according to the rules stated in Section 7.1.

For assistance in implementing a function to generate Intra-LC convergence forms, see Appendix C.3.

11. Future Enhancements

The flexible and modular architecture for FACR described in this document allows Frogans address composition rules to evolve quickly and easily over time, as required by the two-part model presented in the IFAP specification [IFAP] (see IFAP, section 1.4 Stability and Security).

Thus, new versions of this FACR specification can be prepared as needed in order to take into account work that will be contributed to the community in the future concerning international identifiers. This can relate to new or modified rules developed by various organizations to mitigate security issues related to international identifiers.

One or more of the following types of change can be envisaged when a new version of this FACR specification is being prepared:

* Adding a new linguistic category

  One or more linguistic categories can be added to FACR provided that the rules stated in Section 3.1, Section 3.2, Section 4.1, Section 4.2, Section 6, and Section 7 are respected.

  For each linguistic category added, the corresponding languages and writing systems, the employable characters, the arrangement rules, the overlapping linguistic categories, and the Intra-LC convergence form types need to be defined.

* Modifying the rules of a linguistic category

  The rules of a linguistic category concern both its employable characters and its arrangement rules.

  The employable characters of an existing linguistic category can be modified provided that the rules stated in Section 4.1 and Section 6 are respected. Likewise, the arrangement rules of an existing linguistic category can be modified provided that the rules stated in Section 4.2 and Section 6 are respected.

* Modifying or adding a type of Intra-LC convergence form

  There can be one or more types of Intra-LC convergence form defined for a linguistic category.

  One or more types of Intra-LC convergence form can be modified or added for an existing linguistic category provided that the rules stated in Section 3.2 are respected.

- Modifying the type of Inter-LC convergence form

  There is only one type of Inter-LC convergence form.

  The type of Inter-LC convergence form can be modified provided that the rules stated in Section 3.2 are respected.

12. References

12.1. Normative references

[ASCII] American National Standards Institute (formerly United States of America Standards Institute), "USA Code for Information Interchange", ANSI X3.4-1968, 1968.

[CLDR] The Unicode Consortium, "Unicode Common Locale Data Repository", Version 26, September 2014, <http://cldr.unicode.org/index/downloads/cldr-26>.

CLDR XML data files are contained in the core.zip file available for download at the following URL: <http://unicode.org/Public/cldr/26/>

[IDN-CN] IANA Repository of IDN Practices, "IDN Table for the .CN Country-Code Top-Level Domain", Contributed by CNNIC, Version 4.0, March 2005, <http://www.iana.org/domains/idn-tables/tables/cn_zh-cn_4.0.html>.

[IDN-HE] Internet Assigned Numbers Authority (IANA), "Repository of Internationalized Domain Name (IDN) Practices: ISOC-IL", Version 1.0, December 2010, <http://www.iana.org/domains/idn-tables/tables/il_he_1.0.html>.

[IDN-JP] Internet Assigned Numbers Authority (IANA), "Repository of Internationalized Domain Name (IDN) Practices: Japan Registry Services Co., Ltd.", Version 1.2, August 2005, <http://www.iana.org/domains/idn-tables/tables/jp_ja-jp_1.2.html>.

[IDN-TH] Internet Assigned Numbers Authority (IANA), "Repository of Internationalized Domain Name (IDN) Practices: Thailand Network Information Center", Version 1.0, June 2004, <http://www.iana.org/domains/idn-tables/tables/th_th-th_1.0.html>.

[IDN-TW] Internet Assigned Numbers Authority (IANA), "Repository of Internationalized Domain Name (IDN) Practices: Taiwan Network Information Center(TWNIC)", Version 4.0.1, March 2005, <http://www.iana.org/domains/idn-tables/tables/tw_zh-tw_4.0.1.html>.

[IFAP] OP3FT, "International Frogans Address Pattern", Version 1.1, ISBN 978-2-37313-000-3, November 2014, <https://www.frogans.org/en/resources/ifap/access.html>.

[RFC3743] Konishi, K., Huang, K., Qian, H., and Y. Ko, "Joint Engineering Team (JET) Guidelines for Internationalized Domain Names (IDN) Registration and Administration for Chinese, Japanese, and Korean", RFC 3743, April 2004, <http://www.ietf.org/rfc/rfc3743.txt>.

[UAX24] The Unicode Consortium, Davis, M., and K. Whistler, "Unicode Standard Annex #24: Unicode Script Property", Version 7.0.0, Revision 22, June 2014, <http://www.unicode.org/reports/tr24/tr24-22.html>.

[UAX44] The Unicode Consortium, "Unicode Standard Annex #44: Unicode Character Database", Version 7.0.0, Revision 14, June 2014, <http://www.unicode.org/reports/tr44/tr44-14.html>.

[Unicode] The Unicode Consortium, "The Unicode Standard", Version 7.0.0, (Mountain View, CA: The Unicode Consortium, 2014. ISBN 978-1-936213-09-2), June 2014, <http://www.unicode.org/versions/Unicode7.0.0/>.

[UTS35] The Unicode Consortium, Davis, M., and other CLDR committee members, "Unicode Technical Standard #35: Unicode Locale Data Markup Language (LDML)", Version 25, Revision 35, March 2014, <http://www.unicode.org/reports/tr35/tr35-35/tr35.html>.

[UTS39] The Unicode Consortium, Davis, M., and M. Suignard, "Unicode Technical Standard #39: Unicode Security Mechanisms", Version 7.0.0, Revision 9, September 2014, <http://www.unicode.org/reports/tr39/tr39-9.html>.

12.2. Informative references

[BYLAWS] OP3FT, "Bylaws of the French Fonds de Dotation OP3FT, Organization for the Promotion, Protection and Progress of Frogans Technology", March 2012, <https://www.op3ft.org/en/resources/bylaws/access.html>.

[FTUP] OP3FT, "Frogans Technology User Policy", <https://www.frogans.org/en/resources/ftup/access.html>.

[GPCALL] ICANN, "Call for Generation Panels to Develop Root Zone Label Generation Rules", July 2013, <https://www.icann.org/news/announcement-2013-07-11-en>.

[IANA-Repository]
Internet Assigned Numbers Authority (IANA), "Repository of Internationalized Domain Name (IDN) Practices", <http://www.iana.org/domains/idn-tables>.

[IDN-KR] Internet Assigned Numbers Authority (IANA), "Repository of Internationalized Domain Name (IDN) Practices: KRNIC", Version 1.0, March 2004, <http://www.iana.org/domains/idn-tables/tables/kr_ko-kr_1.0.html>.

[IDNA2008]
Klensin, J., "Internationalized Domain Names for Applications (IDNA): Definitions and Document Framework", RFC 5890, August 2010, <http://www.ietf.org/rfc/rfc5890.txt>.

IDNA2008 includes several additional documents: RFC 5891, RFC 5892, RFC 5893, RFC 5894, and RFC 5895.

[ISO15924]
International Organization for Standards, "ISO 15924:2004. Information and documentation -- Codes for the representation of names of scripts", January 2004, <http://www.iso.org/obp/ui/#iso:std:iso:15924:ed-1:v1:en>.

[ISO639] International Organization for Standardization, "Codes for the Representation of Names of Languages", ISO 639, <http://www.iso.org/iso/home/standards/language_codes.htm>.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", March 1997, <http://www.ietf.org/rfc/rfc2119.txt>.

[RFC5564] El-Sherbiny, A., Farah, M., Oueichek, I., and A. Al-Zoman, "Linguistic Guidelines for the Use of the Arabic Language in Internet Domains", RFC 5664, ISSN 2070-1721, February 2010, <http://www.ietf.org/rfc/rfc5564.txt>.

[RFC5892] Falstrom, P., "The Unicode Code Points and Internationalized Domain Names for Applications (IDNA)", RFC 5892, ISSN 2070-1721, August 2010, <http://www.ietf.org/rfc/rfc5892.txt>.

[UDRP-F] OP3FT, "Uniform Dispute Resolution Policy for Frogans Addresses (UDRP-F)", <https://www.frogans.org/en/resources/udrpf/access.html>.

[UTR36] The Unicode Consortium, Davis, M., and M. Suignard, "Unicode Technical Report #36, Unicode Security Considerations", Revision 15, September 2014, <http://www.unicode.org/reports/tr36/tr36-15.html>.

Appendix A. FACR Lookup Tables

This appendix describes the FACR lookup tables used in Appendix C which provides assistance in implementing this specification.

This appendix is not normative. Its contents do not replace the definitions and rules previously set forth in this specification, nor do they define any new rules.

FACR lookup tables are files containing pre-processed lists of code points. This data is provided separately from this specification document in order to make the data easier to use for developers. FACR lookup tables are accessible at the same permanent URL as this specification document (see the first page of this document).

Each FACR lookup table is assigned a unique reference in FLTnn_Label format, where nn is a zero-padded two-digit sequential number and Label is a label where words are separated by the underscore (_) character.

Each FACR lookup table is provided in CSV format. The content of the file has the following characteristics:

* The file is encoded using the ASCII character set [ASCII]. Each line of the file ends with the ASCII character LF.

* The first lines in the file are comments starting with the ASCII character # (number sign). They include the FACR lookup table reference, a brief description of its contents and use, the file name, and the file creation date. The comments also include: the list of third-party source materials and the lists of IFAP and other FACR lookup tables used to create the lookup table; the description of the fields in the lookup table; and the method used to compute the field values in the lookup table.

* The first line of the file that is not a comment contains the field names of the lookup table, in uppercase, separated by the ASCII character comma (,).

* Each subsequent line of the file is a data line containing field values, separated by the ASCII character comma (,).

* The number of fields per data line remains constant. It is possible for a lookup table to contain only one field.

* The name of the first field is CODE_POINT. The value of this field represents either an individual code point or a continuous range of code points. Individual code points are represented in

'cphex' format, and ranges of code points in 'cphex1..cphex2' format, where 'cphex', 'cphex1', and 'cphex2' contain between four and six uppercase hexadecimal digits, and '..' is two consecutive ASCII full stop characters (.). The first and last points of a range are included in the range.

* The next fields contain information related to the code point or range of code points defined in the first field. Any code point included in the value of such fields is represented using the 'cphex' format described above. The value of such fields may be empty on some data lines.

* A code point cannot be listed in the first field of more than one data line, neither as an individual code point nor within a range. The data lines in the file are sorted in increasing order by the code point number of the first field.

* No comments are included between two data lines, at the end of a data line, or at the end of the file.

The remainder of this section lists all the 23 FACR lookup tables used in Appendix C.

See the comments in each lookup table for a brief description of its contents and use.

The hash value provided for each FACR lookup table is computed using the secure hash algorithm SHA-256 of the National Institute of Standards and Technology.

- Reference: FLT01_LC_Latin_Employable

  File name: facr10-adopted.spec.flt01-lc-latin-employable.txt
  File size: 17,270 bytes
  Total number of lines: 461
  Total number of data lines: 061
  File sha256 hash:
  b6111d3371842df0e1f183074661c2869dfc0462bd1c3d5a7692a8064a0b741a

- Reference: FLT02_LC_Chinese_Employable

  File name: facr10-adopted.spec.flt02-lc-chinese-employable.txt
  File size: 29,746 bytes
  Total number of lines: 1,617
  Total number of data lines: 1,346
  File sha256 hash:
  ac37d9898ad1f03f487c7b24fe097fa6f7637d878bae7baf1f851bf4e416c02a

- Reference: FLT03_LC_Japanese_Employable

  File name: facr10-adopted.spec.flt03-lc-japanese-employable.txt
  File size: 55,004 bytes
  Total number of lines: 4,311
  Total number of data lines: 4,044
  File sha256 hash:
  d94d766ba24f057a8977d63162c056186ddfe8103b56579edad19157d034e8f6

- Reference: FLT04_LC_Korean_Employable

  File name: facr10-adopted.spec.flt04-lc-korean-employable.txt
  File size: 23,439 bytes
  Total number of lines: 1,141
  Total number of data lines: 735
  File sha256 hash:
  3e4f20612f36564f655b0e57ddae0019b5625ce3d92f91e8c18e17c7010f8e52

- Reference: FLT05_LC_Arabic_Employable

  File name: facr10-adopted.spec.flt05-lc-arabic-employable.txt
  File size: 16,816 bytes
  Total number of lines: 434
  Total number of data lines: 034
  File sha256 hash:
  008414b14c2bd5c623af237f3c63bc77a10fc8e8a0fbd34eb3efd72d2fd4be05

- Reference: FLT06_LC_Cyrillic_Employable

  File name: facr10-adopted.spec.flt06-lc-cyrillic-employable.txt
  File size: 16,500 bytes
  Total number of lines: 413
  Total number of data lines: 017
  File sha256 hash:
  f42757110eef486fab8bc1087afa0c699992d14eae1202b3140248ccbba54c2f

- Reference: FLT07_LC_Hebrew_Employable

  File name: facr10-adopted.spec.flt07-lc-hebrew-employable.txt
  File size: 8,449 bytes
  Total number of lines: 225
  Total number of data lines: 004
  File sha256 hash:
  a18f18ad9c3dee61c5277b0b5cfe52befed8475d944488d0fab431a5bb2c97dc

- Reference: FLT08_LC_Devanagari_Employable

  File name: facr10-adopted.spec.flt08-lc-devanagari-employable.txt
  File size: 16,498 bytes
  Total number of lines: 411
  Total number of data lines: 015
  File sha256 hash:
  4a45759ecea703998a8a425310fe3fa17cb13d2525d21104ea8cbbff9b0ca173

- Reference: FLT09_LC_Thai_Employable

  File name: facr10-adopted.spec.flt09-lc-thai-employable.txt
  File size: 8,527 bytes
  Total number of lines: 230
  Total number of data lines: 009
  File sha256 hash:
  bd17fd803feb08c312d618befd72ad05f0750a28544ef59e418e3531c270ba3e

- Reference: FLT10_LC_Greek_Employable

  File name: facr10-adopted.spec.flt10-lc-greek-employable.txt
  File size: 16,328 bytes
  Total number of lines: 406
  Total number of data lines: 010
  File sha256 hash:
  42ab1db3712b901609c49259169c8bce25640d6ae8f53de0b13ee6c814884efe

- Reference: FLT11_Decimal_Number_Ranges

  File name: facr10-adopted.spec.flt11-decimal-number-ranges.txt
  File size: 7,295 bytes
  Total number of lines: 211
  Total number of data lines: 049
  File sha256 hash:
  fe9841a96c873679f6caac4d4a8c52b75e03d0c0d37f4cf1f1e0a01ce16106d6

- Reference: FLT12_Intra_LC_Latin_Confusable

  File name: facr10-adopted.spec.flt12-intra-lc-latin-confusable.txt
  File size: 40,692 bytes
  Total number of lines: 2,193
  Total number of data lines: 1,628
  File sha256 hash:
  7f3bfacfe9d304313726712b2157e034e68a42ef8292a1756a02a4194a9f154b

- Reference: FLT13_Intra_LC_Chinese_Confusable

  File name: facr10-adopted.spec.flt13-intra-lc-chinese-confusable.txt
  File size: 39,101 bytes
  Total number of lines: 2,038
  Total number of data lines: 1,472
  File sha256 hash:
  655c10765f82ef594432c34b6b6de67d0c76a0ab9cbf93e42fd25c46a2cbd164

- Reference: FLT14_Intra_LC_Chinese_Variant

  File name: facr10-adopted.spec.flt14-intra-lc-chinese-variant.txt
  File size: 55,141 bytes
  Total number of lines: 4,704
  Total number of data lines: 4,437
  File sha256 hash:
  a7aa6e66cb04969495586c4b8a6c4c975f427fc4ef52dd1fb7e167cefbb6d277

- Reference: FLT15_Intra_LC_Japanese_Confusable

  File name: facr10-adopted.spec.flt15-intra-lc-japanese-confusable.txt
  File size: 39,114 bytes
  Total number of lines: 2,038
  Total number of data lines: 1,472
  File sha256 hash:
  b974b568f96c2c2f6408838489684a027a3cae5834df541ec9d54900f8bfbd29

- Reference: FLT16_Intra_LC_Korean_Confusable

  File name: facr10-adopted.spec.flt16-intra-lc-korean-confusable.txt
  File size: 39,083 bytes
  Total number of lines: 2,038
  Total number of data lines: 1,472
  File sha256 hash:
  24d3a59963fbde6ab4377b7ea09945b254914cb69e5ba4a26e15219ea6b01c69

- Reference: FLT17_Intra_LC_Arabic_Confusable

  File name: facr10-adopted.spec.flt17-intra-lc-arabic-confusable.txt
  File size: 38,738 bytes
  Total number of lines: 2,012
  Total number of data lines: 1,446
  File sha256 hash:
  7944e9c06bc942bc03e8ed3d24ce432ce8dec64b0cd9a140237ab9599346cabb

- Reference: FLT18_Intra_LC_Cyrillic_Confusable

  File name: facr10-adopted.spec.flt18-intra-lc-cyrillic-confusable.txt
  File size: 39,399 bytes
  Total number of lines: 2,065
  Total number of data lines: 1,499
  File sha256 hash:
  012477868abf0fc9a91ab01b562bbd86f5afc40caa174d4af5e7188252d6a751

- Reference: FLT19_Intra_LC_Hebrew_Confusable

  File name: facr10-adopted.spec.flt19-intra-lc-hebrew-confusable.txt
  File size: 38,738 bytes
  Total number of lines: 2,012
  Total number of data lines: 1,446
  File sha256 hash:
  0dbb79aa155e1acbfec116ccd5be5afa39cd1ab5ab5f097a489616a3a79f9be3

- Reference: FLT20_Intra_LC_Devanagari_Confusable

  File name: facr10-adopted.spec.flt20-intra-lc-Devanagari-confusable.txt
  File size: 38,804 bytes
  Total number of lines: 2,013
  Total number of data lines: 1,446
  File sha256 hash:
  a5147bfc15574020d2d966b92e59045a8f520f9da7a5e7faf351825dd1f60317

- Reference: FLT21_Intra_LC_Thai_Confusable

  File name: facr10-adopted.spec.flt21-intra-lc-thai-confusable.txt
  File size: 39,059 bytes
  Total number of lines: 2,037
  Total number of data lines: 1,472
  File sha256 hash:
  6e56d151b5856d4dafee72d2efbd1bad9c8136267f1d70c2f6f5771ddb1b177f

- Reference: FLT22_Intra_LC_Greek_Confusable

  File name: facr10-adopted.spec.flt22-intra-lc-greek-confusable.txt
  File size: 39,118 bytes
  Total number of lines: 2,044
  Total number of data lines: 1,479
  File sha256 hash:
  5902a5425211c2879066fb7d8af15e7ed35be394805c9a15925d93396b959184

- Reference: FLT23_Inter_LC

  File name: facr10-adopted.spec.flt23-inter-lc.txt
  File size: 49,744 bytes
  Total number of lines: 2,912
  Total number of data lines: 2,346
  File sha256 hash:
  2cc158c0b50399ea54d5851c44b4a21fbd7ed78b3dbade0435937b70a26517a8

Appendix B. Pseudocode Syntax

This appendix describes the syntax and conventions for the pseudocode used in Appendix C which provides assistance in implementing this specification.

This appendix is not normative. Its contents do not replace the definitions and rules previously set forth in this specification, nor do they define any new rules.

The pseudocode uses the following syntax and conventions.

All keywords are written in uppercase. The names of all functions, variables, and data objects are written in lowercase.

Spaces are used to separate elements.

Braces ({ and }) are used to delimit blocks of pseudocode.

To improve legibility, the text of the comments is not included in the pseudocode. Instead, comments are referenced by a number between angle brackets (< and >) at the end of a line. For example: <1> indicates comment number 1.

The following statements are used:

* FUNCTION: defines a function. The keyword FUNCTION is followed by the function name, then by a list of one or more parameter names between parentheses.

* VAR: defines a variable used in a function. The VAR keyword is followed by the name of the variable.

* RETURN: exits a function. They keyword RETURN is followed by the value returned by the function.

* CALL: calls a function. The keyword CALL is followed by the name of the called function, then by a list of one or more parameter values between parentheses. The list matches the definition of the called function.

* IF: tests an expression. The IF keyword is followed by the expression between parentheses, then by a block of pseudocode between braces to be executed if the expression evaluates to true.

* ELSE: follows an IF statement. The ELSE keyword is followed either by another IF statement or by a block of pseudocode, which are executed if the expression defined by the previous IF

statement evaluates to false. The pseudocode may contain cascading ELSE statements.

* FOR: defines a loop associated with an index. The FOR keyword is followed by the name of the index, the equal sign (=), the first value included in the index range, the TO keyword, then by the last value included in the index range, then by a block of pseudocode to be executed for each iteration of the loop. If the first or the last value of the index range is defined by an expression, then that expression is included between parentheses. If the last value in the index range is lower than the first value, then the TO keyword is replaced by the DOWNTO keyword. The index is incremented or decremented by one at each iteration of the loop.

* WHILE: defines a loop associated with an expression. The WHILE keyword is followed by the expression between parentheses, then by a block of pseudocode between braces to be executed for each iteration of the loop if the expression evaluates to true. Whenever the expression is evaluated to false, execution continues after the block of pseudocode.

* BREAK: exits a FOR or WHILE loop. The BREAK keyword is not followed by other keywords. Execution continues after the block of pseudocode defined in the loop.

The following logical expressions are used:

* (a == b) tests whether the value of a equals the value of b.

* (a != b) tests whether the value of a is different from the value of b.

* (c OR d) tests whether either of the expressions c or d evaluates to true.

* (c AND d) tests whether both the expressions c and d evaluate to true.

* (NOT c) negates the expression c.

Parentheses are used to combine groups of logical expressions.

The equal sign (=) is used in a block of pseudocode to assign a value to a variable.

The remainder of this section describes two data objects that are specific to the implementation of this specification:

* TABLE: defines a read-only data object containing an IFAP lookup table. For a description of the IFAP lookup table contents, see Appendix A.

* LIST: defines a read/write data object containing a list of code points.

The following methods are defined for a TABLE data object named my_table:

* my_table.CONTAINS (code_point): looks up in my_table a code point with the value of code_point. This method returns either true if a code point with value of code_point is found, or false otherwise.

* my_table.LOOKUP (code_point, field_name): looks up in my_table the value of the field called field_name for the code point equal to the value of code_point. When used in the pseudocode, the name of the field is preceded by the number sign (#). This method returns either the value of the field called field_name for the code point with the value of code_point, or NULL if there is no such code point.

* my_table.FIND (logical_expression): searches in my_table for a code point whose field values match certain conditions defined in the logical expression provided as a parameter. In the logical expression, the names of the fields that the conditions apply to are preceded by the number sign (#). This method returns either the value of a code point meeting the conditions, or NULL if there is no such code point.

The following property and methods are defined for a LIST data object named my_list:

* my_list.COUNT: returns the number of code points in the list

* my_list.GET (i): returns the value of the code point found at index i in the list. The range of index i is from 0 (the first code point) to (my_list.COUNT - 1) (the last code point in the list).

* my_list.APPEND (code_point_series): appends one or more code points to the list. The code points to append are provided as arguments separated by commas.

- my_list.SET (i, code_point): sets the code point found at index i in the list to the value of code_point.

- my_list.REMOVE (i): removes the code point at index i from the list.

Appendix C. Assistance in Implementing the Specification

This appendix provides a series of processes that can be used to implement this specification.

This appendix is not normative. Its contents do not replace the definitions and rules previously set forth in this specification, nor do they define any new rules.

This appendix does not cover the following parts of the specification, as they do not present any particular implementation difficulties: Checking Whether Two Valid Network Names Are Convergent (Section 8), and Checking Whether Two Valid Site Names Are Convergent (Section 9).

Given the limited length of Frogans addresses [IFAP] (see IFAP, section 6), the processes are designed to minimize the size of the FACR lookup tables rather than to optimize process performance.

The four sections in this appendix provide for each function: the function name and description; the functions it is called by and the functions it calls; the FACR lookup tables used by the function; the input parameters; the possible values returned by the function; a numbered list of comments related to the pseudocode; and finally pseudocode describing the function. Comments in the pseudocode are indicated by a number between angle brackets (< and >).

Some functions in these sections call functions provided in the IFAP specification (see IFAP, appendix C) which are identified by including '_ifap_' in the function name.

C.1. Employable Characters

This section provides assistance in implementing a process that verifies whether a candidate string corresponding to a network name or a site name associated with a linguistic category complies with the employable character rules defined in this FACR specification.

One function is required to implement this process:

FUNCTION |facr10-adopted-include-c1-verify-employable-characters|

Description:

This is the main function for this process.

It first selects the FACR lookup table to use according to the linguistic category of the candidate string.

Then it verifies each code point in the candidate string by performing a look-up in the selected FACR lookup table. If any code point in the candidate string is not found, then it is invalid and the entire candidate string is rejected. Otherwise, if all the code point look-ups are successful, then the candidate string is accepted.

Prerequisite:

- The candidate string is a network name or a site name that complies with version 1.1 of the IFAP specification, which is the latest available version at the time this specification is being completed.

Called by:

- none

Calls:

- none

IFAP lookup tables used:

- table_FLT01: FLT01_LC_Latin_Employable
- table_FLT02: FLT02_LC_Chinese_Employable
- table_FLT03: FLT03_LC_Japanese_Employable
- table_FLT04: FLT04_LC_Korean_Employable
- table_FLT05: FLT05_LC_Arabic_Employable
- table_FLT06: FLT06_LC_Cyrillic_Employable
- table_FLT07: FLT07_LC_Hebrew_Employable
- table_FLT08: FLT08_LC_Devanagari_Employable
- table_FLT09: FLT09_LC_Thai_Employable
- table_FLT10: FLT10_LC_Greek_Employable

Input:

- lc: a string identifying the linguistic category of the candidate string
- codepoints: a LIST data object containing code points that represent the candidate string

Returns:

true if the candidate string is accepted, or false otherwise

Comments:

none

Pseudocode:

```
,-------------------------------------------------------------.
| FUNCTION c1_verify_employable_characters (lc, codepoints)   |
| {                                                           |
TABLE table_FLT01
TABLE table_FLT02
TABLE table_FLT03
TABLE table_FLT04
TABLE table_FLT05
TABLE table_FLT06
TABLE table_FLT07
TABLE table_FLT08
TABLE table_FLT09
TABLE table_FLT10
TABLE lookup_table
VAR cur_cp
VAR index
IF (lc == 'LC-Latin')
{
  lookup_table = table_FLT01
}
ELSE IF (lc == 'LC-Chinese')
{
  lookup_table = table_FLT02
}
ELSE IF (lc == 'LC-Japanese')
{
  lookup_table = table_FLT03
}
ELSE IF (lc == 'LC-Korean')
{
  lookup_table = table_FLT04
}
ELSE IF (lc == 'LC-Arabic')
{
  lookup_table = table_FLT05
}
ELSE IF (lc == 'LC-Cyrillic')
{
  lookup_table = table_FLT06
}
ELSE IF (lc == 'LC-Hebrew')
{
  lookup_table = table_FLT07
}
ELSE IF (lc == 'LC-Devanagari')
{
  lookup_table = table_FLT08
}
ELSE IF (lc == 'LC-Thai')
{
  lookup_table = table_FLT09
```

```
         |    }                                                            |
         |    ELSE IF (lc == 'LC-Greek')                                   |
         |    {                                                            |
         |      lookup_table = table_FLT10                                 |
         |    }                                                            |
         |    ELSE                                                         |
         |    {                                                            |
         |      RETURN false                                               |
         |    }                                                            |
         |    FOR index = 0 TO (codepoints.COUNT - 1)                      |
         |    {                                                            |
         |      cur_cp = codepoints.GET (index)                            |
         |      IF (NOT lookup_table.CONTAINS (cur_cp))                    |
         |      {                                                          |
         |        RETURN false                                             |
         |      }                                                          |
         |    }                                                            |
         |    RETURN true                                                  |
         |  }                                                              |
         `-----------------------------------------------------------------'
```

C.2. Arrangement Rules

This section provides assistance in implementing a process that verifies whether a candidate string corresponding to a network name or a site name associated with a linguistic category complies with the arrangement rules defined in this FACR specification.

The functions described in this section are designed to verify the arrangement rules of a network name. These functions can be easily modified to verify the arrangement rules of a site name. For site names, the modifications involve:

* adding a function to check that the same connector character is used in both the site name and the network name, in cases where both the network name and the site name contain a connector character

* adding a function to check that the same range of decimal digits is used in both the site name and the network name, in cases where both the network name and the site name contain one or more decimal digits

* not applying to the site name the arrangement rule concerning native characters

17 functions are required to implement this process:

FUNCTION |c2_verify_arrangement_rules|

Description:

This is the main function for this process.

It calls a function to verify whether the candidate string follows the arrangement rules of the linguistic category provided as input. If the candidate string does not follow the rules, then the candidate string is rejected. Otherwise the candidate string is accepted.

Prerequisite:

- The candidate string must be accepted by the |facr10-adopted-include-c1-verify-employable-characters| function.

Called by:

- none

Calls:

- |c2_verify_arrangement_rules_latin|
- |c2_verify_arrangement_rules_chinese|
- |c2_verify_arrangement_rules_japanese|
- |c2_verify_arrangement_rules_korean|
- |c2_verify_arrangement_rules_arabic|
- |c2_verify_arrangement_rules_cyrillic|
- |c2_verify_arrangement_rules_hebrew|
- |c2_verify_arrangement_rules_devanagari|
- |c2_verify_arrangement_rules_thai|
- |c2_verify_arrangement_rules_greek|

IFAP lookup tables used:

- none

Input:

- lc: a string identifying the linguistic category of the candidate string
- codepoints: a LIST data object containing code points that represent the candidate string

Returns:

true if the candidate string is accepted, or false otherwise

Comments:

none

Pseudocode:

```
,--------------------------------------------------------------.
```

```
FUNCTION c2_verify_arrangement_rules (lc, codepoints)
{
  VAR res
  IF (lc == 'LC-Latin')
  {
    res = CALL c2_verify_arrangement_rules_latin
                                            (codepoints)
  }
  ELSE IF (lc == 'LC-Chinese')
  {
    res = CALL c2_verify_arrangement_rules_chinese
                                            (codepoints)
  }
  ELSE IF (lc == 'LC-Japanese')
  {
    res = CALL c2_verify_arrangement_rules_japanese
                                            (codepoints)
  }
  ELSE IF (lc == 'LC-Korean')
  {
    res = CALL c2_verify_arrangement_rules_korean
                                            (codepoints)
  }
  ELSE IF (lc == 'LC-Arabic')
  {
    res = CALL c2_verify_arrangement_rules_arabic
                                            (codepoints)
  }
  ELSE IF (lc == 'LC-Cyrillic')
  {
    res = CALL c2_verify_arrangement_rules_cyrillic
                                            (codepoints)
  }
  ELSE IF (lc == 'LC-Hebrew')
  {
    res = CALL c2_verify_arrangement_rules_hebrew
                                            (codepoints)
  }
  ELSE IF (lc == 'LC-Devanagari')
  {
    res = CALL c2_verify_arrangement_rules_devanagari
                                            (codepoints)
  }
  ELSE IF (lc == 'LC-Thai')
  {
    res = CALL c2_verify_arrangement_rules_thai
                                            (codepoints)
  }
```

```
      |   ELSE IF (lc == 'LC-Greek')                                 |
      |   {                                                          |
      |     res = CALL c2_verify_arrangement_rules_greek             |
      |                                                (codepoints)  |
      |   }                                                          |
      |   ELSE                                                       |
      |   {                                                          |
      |     res = false                                              |
      |   }                                                          |
      |   RETURN res                                                 |
      | }                                                            |
      `--------------------------------------------------------------'
```

FUNCTION |c2_verify_arrangement_rules_latin|

Description:

This is a sub-function of the arrangement rules verification process.

It verifies whether the candidate string meets the arrangement rules for LC-Latin.

First it checks whether the candidate string follows the arrangement rule concerning the use of different connector characters. If it does not follow that rule, then the candidate string is rejected.

Otherwise, it checks whether the candidate string follows the arrangement rule concerning the middle dot character. If it does not follow that rule, then the candidate string is rejected.

Otherwise the candidate string is accepted.

Called by:
- c2_verify_arrangement_rules

Calls:
- c2_verify_arrangement_rule_connectors
- c2_verify_arrangement_rule_middle_dot

IFAP lookup tables used:
- none

Input:
- codepoints: a LIST data object containing code points that represent the candidate string

Returns:
true if the candidate string is accepted, or false otherwise

Comments:
none

Pseudocode:

```
,--------------------------------------------------------------.
| FUNCTION c2_verify_arrangement_rules_latin (codepoints)      |
| {                                                            |
|   IF (CALL c2_verify_arrangement_rule_connectors             |
|                                   (codepoints) == false)     |
|   {                                                          |
|     RETURN false                                             |
|   }                                                          |
|   IF (CALL c2_verify_arrangement_rule_middle_dot             |
|                                   (codepoints) == false)     |
|   {                                                          |
|     RETURN false                                             |
|   }                                                          |
|   RETURN true                                                |
| }                                                            |
`--------------------------------------------------------------'
```

FUNCTION |c2_verify_arrangement_rule_connectors|

Description:
This is a sub-function of the arrangement rules verification process.

This function checks whether the candidate string contains two different connector characters.

It analyzes each code point in the candidate string to find connector characters.

If the candidate string contains two connector characters that are not identical, then the candidate string is rejected.

Otherwise the candidate string is accepted.

Called by:
- c2_verify_arrangement_rules_latin
- c2_verify_arrangement_rules_japanese

Calls:

- c2_ifap_is_connector_character

IFAP lookup tables used:

- none

Input:

- codepoints: a LIST data object containing code points that represent the candidate string

Returns:

true if the candidate string is accepted, or false otherwise

Comments:

none

Pseudocode:

```
,-------------------------------------------------------------.
| FUNCTION c2_verify_arrangement_rule_connectors (codepoints) |
| {                                                           |
|   VAR cur_cp                                                |
|   VAR index                                                 |
|   VAR first_connector                                       |
|   first_connector = NULL                                    |
|   FOR index = 0 TO (codepoints.COUNT - 1)                   |
|   {                                                         |
|     cur_cp = codepoints.GET (index)                         |
|     IF (CALL c2_ifap_is_connector_character (cur_cp)        |
|                                                    == true) |
|     {                                                       |
|       IF (first_connector == NULL)                          |
|       {                                                     |
|         first_connector = cur_cp                            |
|       }                                                     |
|       ELSE                                                  |
|       {                                                     |
|         IF (cur_cp != first_connector)                      |
|         {                                                   |
|           RETURN false                                      |
|         }                                                   |
|       }                                                     |
|     }                                                       |
|   }                                                         |
|   RETURN true                                               |
| }                                                           |
`-------------------------------------------------------------'
```

FUNCTION |c2_ifap_is_connector_character|

Description:

This is a sub-function of the arrangement rules verification process.

This function checks whether a code point represents a connector character, according to version 1.1 of the IFAP specification.

Sample pseudocode to implement this function is provided in Appendix C.5, FUNCTION |c5_is_connector_character|, of version 1.1 of the IFAP specification.

Called by:

- c2_verify_arrangement_rule_connectors

Input:

- a_codepoint: a code point.

Returns:

true if a_codepoint represents a connector character, or false otherwise.

FUNCTION |c2_verify_arrangement_rule_middle_dot|

Description:

This is a sub-function of the arrangement rules verification process.

This function checks whether the candidate string contains the U+00B7 MIDDLE DOT character, and if so, whether it complies with the arrangement rules for the U+00B7 MIDDLE DOT character defined in Section 10.1.3 of the FACR specification.

It searches for the required two code points, U+004C LATIN CAPITAL LETTER L and U+006C LATIN SMALL LETTER L, which surround the U+00B7 MIDDLE DOT code point. If the required code points are not found, then the candidate string is rejected.

Otherwise the candidate string is accepted.

Called by:

- c2_verify_arrangement_rules_latin

Calls:

- none

IFAP lookup tables used:

- none

Input:

- codepoints: a LIST data object containing code points that represent the candidate string

Returns:

true if the candidate string is accepted, or false otherwise

Comments:

none

Pseudocode:

```
         ,-------------------------------------------------------------.
         | FUNCTION c2_verify_arrangement_rule_middle_dot (codepoints) |
         | {                                                           |
         |   VAR cur_cp                                                |
         |   VAR prev_cp                                               |
         |   VAR next_cp                                               |
         |   VAR index                                                 |
         |   VAR first_connector                                       |
         |   first_connector = NULL                                    |
         |   FOR index = 0 TO (codepoints.COUNT - 1)                   |
         |   {                                                         |
         |     cur_cp = codepoints.GET (index)                         |
         |     IF (cur_cp == U+00B7)                                   |
         |     {                                                       |
         |       IF ((index == 0) OR                                   |
         |           (index == codepoints.COUNT - 1))                  |
         |       {                                                     |
         |         RETURN false                                        |
         |       }                                                     |
         |       prev_cp = codepoints.GET (index - 1)                  |
         |       next_cp = codepoints.GET (index + 1)                  |
         |       IF ((prev_cp != U+006C) AND                           |
         |           (prev_cp != U+004C))                              |
         |       {                                                     |
         |         RETURN false                                        |
         |       }                                                     |
         |       IF (next_cp != prev_cp)                               |
         |       {                                                     |
         |         RETURN false                                        |
         |       }                                                     |
|     }                                                       |
|   }                                                         |
|   RETURN true                                               |
| }                                                           |
`-------------------------------------------------------------'
```

FUNCTION |c2_verify_arrangement_rules_chinese|

Description:
This is a sub-function of the arrangement rules verification process.

It verifies whether the candidate string meets the arrangement rules for LC-Chinese.

It checks whether the candidate string follows the native arrangement rule which determines that the candidate string contains at least one code point representing a Han character. If it does not follow that rule, then the candidate string is rejected.

Otherwise the candidate string is accepted.

Called by:
- c2_verify_arrangement_rules

Calls:
- c2_verify_arrangement_rule_native

IFAP lookup tables used:
- none

Input:
- codepoints: a LIST data object containing code points that represent the candidate string

Returns:
true if the candidate string is accepted, or false otherwise

Comments:
none

Pseudocode:

```
,-------------------------------------------------------------.
| FUNCTION c2_verify_arrangement_rules_chinese (codepoints)   |
| {                                                           |
|   IF (CALL c2_verify_arrangement_rule_native                |
   |                         ('LC-Chinese', codepoints) == false)  |
   |   {                                                             |
   |     RETURN false                                                |
   |   }                                                             |
   |   RETURN true                                                   |
   | }                                                               |
   `---------------------------------------------------------------'
```

FUNCTION |c2_verify_arrangement_rule_native|

Description:

This is a sub-function of the arrangement rules verification process.

It first selects the FACR lookup table to use according to the linguistic category provided as input.

Then it checks the script of each code point in the candidate string by performing a look-up in the selected FACR lookup table. The look-up returns the value of SCRIPT for each code point.

If the candidate string does not contain at least one code point with the Unicode Script property equal to the value of the Script property for the linguistic category, then the candidate string is rejected.

Otherwise the candidate string is accepted.

Called by:

- c2_verify_arrangement_rules_chinese
- c2_verify_arrangement_rules_japanese
- c2_verify_arrangement_rules_korean
- c2_verify_arrangement_rules_thai

Calls:

- none

IFAP lookup tables used:

- table_FLT02: FLT02_LC_Chinese_Employable
- table_FLT03: FLT03_LC_Japanese_Employable
- table_FLT04: FLT04_LC_Korean_Employable
- table_FLT09: FLT09_LC_Thai_Employable

Input:

- lc: a string identifying the linguistic category of the candidate string

- codepoints: a LIST data object containing code points that represent the candidate string

Returns:
true if the candidate string is accepted, or false otherwise

Comments:
none

Pseudocode:

```
,-------------------------------------------------------------.
| FUNCTION c2_verify_arrangement_rule_native (lc, codepoints) |
| {                                                           |
|   TABLE table_FLT02                                         |
|   TABLE table_FLT03                                         |
|   TABLE table_FLT04                                         |
|   TABLE table_FLT09                                         |
|   TABLE lookup_table                                        |
|   VAR index                                                 |
|   VAR cur_cp                                                |
|   VAR script                                                |
|   IF (lc == 'LC-Chinese')                                   |
|   {                                                         |
|     lookup_table = table_FLT02                              |
|   }                                                         |
|   ELSE IF (lc == 'LC-Japanese')                             |
|   {                                                         |
|     lookup_table = table_FLT03                              |
|   }                                                         |
|   ELSE IF (lc == 'LC-Korean')                               |
|   {                                                         |
|     lookup_table = table_FLT04                              |
|   }                                                         |
|   ELSE IF (lc == 'LC-Thai')                                 |
|   {                                                         |
|     lookup_table = table_FLT09                              |
|   }                                                         |
|   ELSE                                                      |
|   {                                                         |
|     RETURN false                                            |
|   }                                                         |
|   FOR index = 0 TO (codepoints.COUNT - 1)                   |
|   {                                                         |
|     cur_cp = codepoints.GET (index)                         |
|     script = lookup_table.LOOKUP (cur_cp, #script)          |
|     IF (script == NULL)                                     |
|     {                                                       |
         |       RETURN false                                          |
         |     }                                                       |
         |     IF (lc == 'LC-Chinese')                                 |
         |     {                                                       |
         |       IF (script == 'Han')                                  |
         |       {                                                     |
         |         RETURN true                                         |
         |       }                                                     |
         |     }                                                       |
         |     ELSE IF (lc == 'LC-Japanese')                           |
         |     {                                                       |
         |       IF ((script == 'Han') OR                              |
         |           (script == 'Katakana') OR                         |
         |           (script == 'Hiragana'))                           |
         |       {                                                     |
         |         RETURN true                                         |
         |       }                                                     |
         |     }                                                       |
         |     ELSE IF (lc == 'LC-Korean')                             |
         |     {                                                       |
         |       IF ((script == 'Hangul') OR                           |
         |           (script == 'Han'))                                |
         |       {                                                     |
         |         RETURN true                                         |
         |       }                                                     |
         |     }                                                       |
         |     ELSE IF (lc == 'LC-Thai')                               |
         |     {                                                       |
         |       IF (script == 'Thai')                                 |
         |       {                                                     |
         |         RETURN true                                         |
         |       }                                                     |
         |     }                                                       |
         |   }                                                         |
         |   RETURN false                                              |
         | }                                                           |
         `-------------------------------------------------------------'
```

FUNCTION |c2_verify_arrangement_rules_japanese|

Description:

This is a sub-function of the arrangement rules verification process.

It verifies whether the candidate string meets the arrangement rules for LC-Japanese.

First it checks whether the candidate string follows the arrangement rule concerning the use of different connector characters. If it does not follow that rule, then the candidate string is rejected.

It checks whether the candidate string follows the native arrangement rule which determines that the candidate string contains at least one code point representing a Hiragana, Katakana or Kanji character. If it does not follow that rule, then the candidate string is rejected.

Otherwise, it checks whether the candidate string follows the arrangement rule concerning the Katakana middle dot character. If it does not follow that rule, then the candidate string is rejected.

Otherwise the candidate string is accepted.

Called by:

- c2_verify_arrangement_rules

Calls:

- c2_verify_arrangement_rule_connectors
- c2_verify_arrangement_rule_native
- c2_verify_arrangement_rule_katakana_middle_dot

IFAP lookup tables used:

- none

Input:

- codepoints: a LIST data object containing code points that represent the candidate string

Returns:
true if the candidate string is accepted, or false otherwise

Comments:
none

Pseudocode:

```
,--------------------------------------------------------------.
| FUNCTION c2_verify_arrangement_rules_japanese (codepoints)   |
| {                                                            |
|   IF (CALL c2_verify_arrangement_rule_connectors             |
|                                    (codepoints) == false)    |
|   {                                                          |
|     RETURN false                                             |
         |   }                                                         |
         |   IF (CALL c2_verify_arrangement_rule_native                |
         |                      ('LC-Japanese', codepoints) == false)  |
         |   {                                                         |
         |     RETURN false                                            |
         |   }                                                         |
         |   IF (CALL c2_verify_arrangement_rule_katakana_middle_dot   |
         |                      ('LC-Japanese', codepoints) == false)  |
         |   {                                                         |
         |     RETURN false                                            |
         |   }                                                         |
         |   RETURN true                                               |
         | }                                                           |
         `-------------------------------------------------------------'
```

FUNCTION |c2_verify_arrangement_rule_katakana_middle_dot|

Description:

This is a sub-function of the arrangement rules verification process.

This function checks whether the candidate string contains the U+30FB KATAKANA MIDDLE DOT character, and if so, whether it complies with the arrangement rules for the U+30FB KATAKANA MIDDLE DOT character defined in Section 10.3.3 of the FACR specification.

It searches for the two code points which surround the U+30FB KATAKANA MIDDLE DOT code point. If these code points do not belong to the Katakana, Hiragana or Han script, then the candidate string is rejected.

Otherwise the candidate string is accepted.

Called by:

- c2_verify_arrangement_rules_japanese

Calls:

- none

IFAP lookup tables used:

- table_FLT03: FLT03_LC_Japanese_Employable

Input:

- lc: a string identifying the linguistic category of the candidate string

- codepoints: a LIST data object containing code points that represent the candidate string

Returns:

true if the candidate string is accepted, or false otherwise

Comments:

none

Pseudocode:

```
,-------------------------------------------------------------.
| FUNCTION c2_verify_arrangement_rule_katakana_middle_dot     |
|                                         (lc, codepoints)    |
| {                                                           |
|   TABLE table_FLT03                                         |
|   TABLE lookup_table                                        |
|   VAR index                                                 |
|   VAR cur_cp                                                |
|   VAR prev_cp                                               |
|   VAR next_cp                                               |
|   VAR script                                                |
|   IF (lc == 'LC-Japanese')                                  |
|   {                                                         |
|     lookup_table = table_FLT03                              |
|   }                                                         |
|   ELSE                                                      |
|   {                                                         |
|     RETURN false                                            |
|   }                                                         |
|   FOR index = 0 TO (codepoints.COUNT - 1)                   |
|   {                                                         |
|     cur_cp = codepoints.GET (index)                         |
|     IF (cur_cp == U+30FB)                                   |
|     {                                                       |
|       IF ((index == 0) OR                                   |
|           (index == codepoints.COUNT - 1))                  |
|       {                                                     |
|         RETURN false                                        |
|       }                                                     |
|       prev_cp = codepoints.GET (index - 1)                  |
|       next_cp = codepoints.GET (index + 1)                  |
|       script = lookup_table.LOOKUP (prev_cp, #script)       |
|       IF (script == NULL)                                   |
|       {                                                     |
|         RETURN false                                        |
|       }                                                     |
|       IF ((script != 'Katakana') AND                        |
         |           (script != 'Hiragana') AND                         |
         |           (script != 'Han'))                                 |
         |       {                                                      |
         |         RETURN false                                         |
         |       }                                                      |
         |       script = lookup_table.LOOKUP (next_cp, #script)        |
         |       IF (script == NULL)                                    |
         |       {                                                      |
         |         RETURN false                                         |
         |       }                                                      |
         |       IF ((script != 'Katakana') AND                         |
         |           (script != 'Hiragana') AND                         |
         |           (script != 'Han'))                                 |
         |       {                                                      |
         |         RETURN false                                         |
         |       }                                                      |
         |     }                                                        |
         |   }                                                          |
         |   RETURN true                                                |
         | }                                                            |
         `--------------------------------------------------------------'
```

FUNCTION |c2_verify_arrangement_rules_korean|

Description:

This is a sub-function of the arrangement rules verification process.

It verifies whether the candidate string meets the arrangement rules for LC-Korean.

First it checks whether the candidate string follows the native arrangement rule which determines that the candidate string contains at least one code point representing a Hangul or Hanja character. If it does not follow that rule, then the candidate string is rejected.

Otherwise the candidate string is accepted.

Called by:

- c2_verify_arrangement_rules

Calls:

- c2_verify_arrangement_rule_native

IFAP lookup tables used:

- none

Input:

- codepoints: a LIST data object containing code points that represent the candidate string

Returns:

true if the candidate string is accepted, or false otherwise

Comments:

none

Pseudocode:

```
,-------------------------------------------------------------.
| FUNCTION c2_verify_arrangement_rules_korean (codepoints)    |
| {                                                           |
|   IF (CALL c2_verify_arrangement_rule_native                |
|                        ('LC-Korean', codepoints) == false)  |
|   {                                                         |
|     RETURN false                                            |
|   }                                                         |
|   RETURN true                                               |
| }                                                           |
`-------------------------------------------------------------'
```

FUNCTION |c2_verify_arrangement_rules_arabic|

Description:

This is a sub-function of the arrangement rules verification process.

It verifies whether the candidate string meets the arrangement rules for LC-Arabic.

It checks whether the candidate string follows the arrangement rule concerning decimal digits. If it does not follow that rule, then the candidate string is rejected.

Otherwise the candidate string is accepted.

Called by:

- c2_verify_arrangement_rules

Calls:

- c2_verify_arrangement_rule_decimal_digits

IFAP lookup tables used:

- none

Input:

- codepoints: a LIST data object containing code points that represent the candidate string

Returns:

true if the candidate string is accepted, or false otherwise

Comments:

none

Pseudocode:

```
,--------------------------------------------------------------.
| FUNCTION c2_verify_arrangement_rules_arabic (codepoints)     |
| {                                                            |
|   IF (CALL c2_verify_arrangement_rule_decimal_digits         |
|                                    (codepoints) == false)    |
|   {                                                          |
|     RETURN false                                             |
|   }                                                          |
|   RETURN true                                                |
| }                                                            |
`--------------------------------------------------------------'
```

FUNCTION |c2_verify_arrangement_rule_decimal_digits|

Description:

This is a sub-function of the arrangement rules verification process.

This function checks whether all the code points in the candidate string that represent a decimal number belong to the same numbering system. If any code point in the candidate string that represents a decimal number does not belong to the same numbering system, then the candidate string is rejected.

Otherwise the candidate string is accepted.

Called by:

- c2_verify_arrangement_rules_arabic
- c2_verify_arrangement_rules_devanagari
- c2_verify_arrangement_rules_thai

Calls:

- none

IFAP lookup tables used:

- table_FLT11: FLT11_Decimal_Number_Ranges

Input:

- codepoints: a LIST data object containing code points that represent the candidate string

Returns:

true if the candidate string is accepted, or false otherwise

Comments:

none

Pseudocode:

```
,--------------------------------------------------------------.
| FUNCTION c2_verify_arrangement_rule_decimal_digits           |
|                                         (codepoints)         |
| {                                                            |
|   TABLE table_FLT11                                          |
|   VAR cur_cp                                                 |
|   VAR index                                                  |
|   VAR range                                                  |
|   VAR first_range                                            |
|   first_range = NULL                                         |
|   FOR index = 0 TO (codepoints.COUNT - 1)                    |
|   {                                                          |
|     cur_cp = codepoints.GET (index)                          |
|     range = table_FLT11.LOOKUP (cur_cp, #range_ref)          |
|     IF (range != NULL)                                       |
|     {                                                        |
|       IF (first_range == NULL)                               |
|       {                                                      |
|         first_range = range                                  |
|       }                                                      |
|       ELSE                                                   |
|       {                                                      |
|         IF (range != first_range)                            |
|         {                                                    |
|           RETURN false                                       |
|         }                                                    |
|       }                                                      |
|     }                                                        |
|   }                                                          |
|   RETURN true                                                |
| }                                                           |
`-------------------------------------------------------------'
```

FUNCTION |c2_verify_arrangement_rules_cyrillic|

Description:

This is a sub-function of the arrangement rules verification process.

LC-Cyrillic does not have any arrangement rules; all candidate strings are accepted.

Called by:

- c2_verify_arrangement_rules

Calls:

- none

IFAP lookup tables used:

- none

Input:

- codepoints: a LIST data object containing code points that represent the candidate string

Returns:

true if the candidate string is accepted, or false otherwise

Comments:

none

Pseudocode:

```
,-------------------------------------------------------------.
| FUNCTION c2_verify_arrangement_rules_cyrillic (codepoints)  |
| {                                                           |
|   RETURN true                                               |
| }                                                           |
`-------------------------------------------------------------'
```

FUNCTION |c2_verify_arrangement_rules_hebrew|

Description:

This is a sub-function of the arrangement rules verification process.

LC-Hebrew does not have any arrangement rules; all candidate strings are accepted.

Called by:

- c2_verify_arrangement_rules

Calls:

- none

IFAP lookup tables used:

- none

Input:

- codepoints: a LIST data object containing code points that represent the candidate string

Returns:

true if the candidate string is accepted, or false otherwise

Comments:

none

Pseudocode:

```
,--------------------------------------------------------------.
| FUNCTION c2_verify_arrangement_rules_hebrew (codepoints)     |
| {                                                            |
|   RETURN true                                                |
| }                                                            |
`--------------------------------------------------------------'
```

FUNCTION |c2_verify_arrangement_rules_devanagari|

Description:

This is a sub-function of the arrangement rules verification process.

It verifies whether the candidate string meets the arrangement rules for LC-Devangari.

It checks whether the candidate string follows the arrangement rule concerning decimal digits. If it does not follow that rule, then the candidate string is rejected.

Otherwise the candidate string is accepted.

Called by:

- c2_verify_arrangement_rules

Calls:

- c2_verify_arrangement_rule_decimal_digits

IFAP lookup tables used:

- none

Input:

- codepoints: a LIST data object containing code points that represent the candidate string

Returns:

true if the candidate string is accepted, or false otherwise

Comments:

none

Pseudocode:

```
,-------------------------------------------------------------.
| FUNCTION c2_verify_arrangement_rules_devanagari (codepoints)|
| {                                                           |
|   IF (CALL c2_verify_arrangement_rule_decimal_digits        |
|                                    (codepoints) == false)   |
|   {                                                         |
|     RETURN false                                            |
|   }                                                         |
|   RETURN true                                               |
| }                                                           |
`-------------------------------------------------------------'
```

FUNCTION |c2_verify_arrangement_rules_thai|

Description:

This is a sub-function of the arrangement rules verification process.

It verifies whether the candidate string meets the arrangement rules for LC-Thai.

It checks whether the candidate string follows the native arrangement rule which determines that the candidate string contains at least one code point representing a Thai character. If it does not follow that rule, then the candidate string is rejected.

It checks whether the candidate string follows the arrangement rule concerning decimal digits. If it does not follow that rule, then the candidate string is rejected.

Otherwise the candidate string is accepted.

Called by:

- c2_verify_arrangement_rules

Calls:

- c2_verify_arrangement_rule_native
- c2_verify_arrangement_rule_decimal_digits

IFAP lookup tables used:

- none

Input:

- codepoints: a LIST data object containing code points that represent the candidate string

Returns:

true if the candidate string is accepted, or false otherwise

Comments:

none

Pseudocode:

```
,-------------------------------------------------------------.
| FUNCTION c2_verify_arrangement_rules_thai (codepoints)      |
| {                                                           |
|   IF (CALL c2_verify_arrangement_rule_native                |
|                          ('LC-Thai', codepoints) == false)  |
|   {                                                         |
|     RETURN false                                            |
|   }                                                         |
|   IF (CALL c2_verify_arrangement_rule_decimal_digits        |
|                                     (codepoints) == false)  |
|   {                                                         |
|     RETURN false                                            |
|   }                                                         |
|    RETURN true                                              |
| }                                                           |
`-------------------------------------------------------------'
```

FUNCTION |c2_verify_arrangement_rules_greek|

Description:

This is a sub-function of the arrangement rules verification process.

LC-Greek does not have any arrangement rules; all candidate strings are accepted.

Called by:

- c2_verify_arrangement_rules

Calls:

- none

IFAP lookup tables used:

- none

Input:

- codepoints: a LIST data object containing code points that represent the candidate string

Returns:

true if the candidate string is accepted, or false otherwise

Comments:

none

Pseudocode:

```
,------------------------------------------------------------.
| FUNCTION c2_verify_arrangement_rules_greek (codepoints)     |
| {                                                           |
|   RETURN true                                               |
| }                                                           |
`------------------------------------------------------------'
```

C.3. Intra-LC Convergence Forms

This section provides assistance in implementing a process that generates a convergence form for a candidate string corresponding to a network name or a site name associated with a linguistic category.

Three functions are required to implement this process:

FUNCTION |c3_generate_intra_lc_convergence_form|

Description:

This is the main function for this process.

It generates the Intra-LC convergence form of a candidate string for a given Intra-LC convergence form type.

It first selects the FACR lookup table to use according to the required Intra-LC convergence form type of the candidate string.

Then it calls a function which uses this table to generate the convergence form of the candidate string.

Prerequisite:

- The candidate string must be accepted by the |c2_verify_arrangement_rules| function.

Called by:

- none

Calls:

- c3_generate_convergence_form

IFAP lookup tables used:

- table_FLT12: FLT12_Intra_LC_Latin_Confusable
- table_FLT13: FLT13_Intra_LC_Chinese_Confusable
- table_FLT14: FLT14_Intra_LC_Chinese_Variant
- table_FLT15: FLT15_Intra_LC_Japanese_Confusable
- table_FLT16: FLT16_Intra_LC_Korean_Confusable
- table_FLT17: FLT17_Intra_LC_Arabic_Confusable
- table_FLT18: FLT18_Intra_LC_Cyrillic_Confusable
- table_FLT19: FLT19_Intra_LC_Hebrew_Confusable
- table_FLT20: FLT20_Intra_LC_Devanagari_Confusable
- table_FLT21: FLT21_Intra_LC_Thai_Confusable
- table_FLT22: FLT22_Intra_LC_Greek_Confusable

Input:

- codepoints: a LIST data object containing code points that represent the candidate string
- cvft: a string identifying the Intra-LC convergence form type

Returns:

a string of Unicode characters representing the Intra-LC convergence form of a candidate string

Comments:

none

Pseudocode:

```
,-------------------------------------------------------------.
| FUNCTION c3_generate_intra_lc_convergence_form (codepoints, |
|                                                  cvft)       |
{
  TABLE table_FLT12
  TABLE table_FLT13
  TABLE table_FLT14
  TABLE table_FLT15
  TABLE table_FLT16
  TABLE table_FLT17
  TABLE table_FLT18
  TABLE table_FLT19
  TABLE table_FLT20
  TABLE table_FLT21
  TABLE table_FLT22
  TABLE lookup_table
  VAR res
  IF (cvft == 'Intra-LC-Latin-Confusable')
  {
    lookup_table = table_FLT12
  }
  ELSE IF (cvft == 'Intra-LC-Chinese-Confusable')
  {
    lookup_table = table_FLT13
  }
  ELSE IF (cvft == 'Intra-LC-Chinese-Variant')
  {
    lookup_table = table_FLT14
  }
  ELSE IF (cvft == 'Intra-LC-Japanese-Confusable')
  {
    lookup_table = table_FLT15
  }
  ELSE IF (cvft == 'Intra-LC-Korean-Confusable')
  {
    lookup_table = table_FLT16
  }
  ELSE IF (cvft == 'Intra-LC-Arabic-Confusable')
  {
    lookup_table = table_FLT17
  }
  ELSE IF (cvft == 'Intra-LC-Cyrillic-Confusable')
  {
    lookup_table = table_FLT18
  }
  ELSE IF (cvft == 'Intra-LC-Hebrew-Confusable')
  {
    lookup_table = table_FLT19
  }
  ELSE IF (cvft == 'Intra-LC-Devanagari-Confusable')
  {
         |     lookup_table = table_FLT20                              |
         |   }                                                         |
         |   ELSE IF (cvft == 'Intra-LC-Thai-Confusable')              |
         |   {                                                         |
         |     lookup_table = table_FLT21                              |
         |   }                                                         |
         |   ELSE IF (cvft == 'Intra-LC-Greek-Confusable')             |
         |   {                                                         |
         |     lookup_table = table_FLT22                              |
         |   }                                                         |
         |   ELSE                                                      |
         |   {                                                         |
         |     RETURN NULL                                             |
         |   }                                                         |
         |   res = CALL c3_generate_convergence_form (codepoints,      |
         |                                            lookup_table)    |
         |   RETURN res                                                |
         | }                                                           |
         `-------------------------------------------------------------'
```

FUNCTION |c3_generate_convergence_form|

Description:

This is a sub-function of the Intra-LC and the Inter-LC convergence forms generation processes.

It generates the convergence form of a candidate string.

First it applies the NFD normalization process to the input candidate string.

Then, by performing a look-up in the selected FACR lookup table, each code point of the NFD normalized candidate string is mapped to the corresponding code point defined in the lookup table for the convergence form type.

Then it applies the NFD normalization process to the transformed candidate string.

Note that for the Intra-LC-Chinese-Variant convergence form type, it is not necessary to apply the NFD normalization process. However, this has no effect on the string returned by the function.

Called by:

- c3_generate_intra_lc_convergence_form

- c4_generate_inter_lc_convergence_form

Calls:

- c3_ifap_normalize_nfd

IFAP lookup tables used:

- none

Input:

- codepoints: a LIST data object containing code points that represent the candidate string
- lookup_table: a string containing the name of the lookup table of the convergence form type

Returns:

a string containing the convergence form of a candidate string

Comments:

none

Pseudocode:

```
,-------------------------------------------------------------.
| FUNCTION c3_generate_convergence_form (codepoints,          |
|                                       lookup_table)         |
| {                                                           |
|   LIST res                                                  |
|   LIST work_nfd_cps                                         |
|   LIST work_mapping_cps                                     |
|   LIST cur_mapping                                          |
|   VAR cur_cp                                                |
|   VAR index                                                 |
|   res = NULL                                                |
|   work_nfd_cps = CALL c3_ifap_normalize_nfd (codepoints)    |
|   FOR index = 0 TO (work_nfd_cps.COUNT - 1)                 |
|   {                                                         |
|     cur_cp =  work_nfd_cps.GET (index)                      |
|     cur_mapping = lookup_table.LOOKUP                       |
|                           (cur_cp, #convergence_mapping)    |
|     IF (cur_mapping == NULL)                                |
|     {                                                       |
|       work_mapping_cps.APPEND (cur_cp)                      |
|     }                                                       |
|     ELSE                                                    |
|     {                                                       |
|       work_mapping_cps.APPEND (cur_mapping)                 |
|     }                                                       |
|   }                                                         |
|   res = CALL c3_ifap_normalize_nfd (work_mapping_cps)        |
|   RETURN res                                                 |
| }                                                            |
`--------------------------------------------------------------'
```

FUNCTION |c3_ifap_normalize_nfd|

Description:

This is a sub-function of the process for generating the convergence form.

The function applies a two-step procedure to generate an NFD normalized string from an input string of code points, according to version 1.1 of the IFAP specification.

Sample pseudocode to implement this function is provided in Appendix C.6, FUNCTION |c6_normalize_nfd|, of version 1.1 of the IFAP specification.

Called by:

- |c3_generate_convergence_form|

Input:

- codepoints: a LIST data object containing code points representing the string to be normalized

Returns:

the NFD normalized string

C.4. Inter-LC Convergence Form

This section provides assistance in implementing a process that generates a convergence form for a candidate string corresponding to a network name.

One function is required to implement this process:

FUNCTION |c4_generate_inter_lc_convergence_form|

Description:

This is the main function for this process.

It generates the Inter-LC convergence form of a candidate string for the Inter-LC convergence form type.

It first sets the FACR lookup table to table_FLT23: FLT23_Inter_LC.

Then it calls a function which uses this table to generate the convergence form of the candidate string.

Prerequisite:

- The candidate string must be accepted by the |c2_verify_arrangement_rules| function.

Called by:

- none

Calls:

- c3_generate_convergence_form

IFAP lookup tables used:

- table_FLT23: FLT23_Inter_LC

Input:

- codepoints: a LIST data object containing code points that represent the candidate string

Returns:

a string of Unicode characters representing the confusable-based Inter-LC convergence form of a candidate string

Comments:

none

Pseudocode:

```
,--------------------------------------------------------------.
| FUNCTION c4_generate_inter_lc_convergence_form (codepoints) |
| {                                                            |
|   TABLE table_FLT23                                          |
|   VAR res                                                    |
|   res = CALL c3_generate_convergence_form (codepoints,       |
|                                            table_FLT23)      |
|   RETURN res                                                 |
| }                                                            |
`--------------------------------------------------------------'
```