Constructed Language (Conlang) JSON Document Specification

Ronald B. Oakes

ron@ron-oakes.us

February 24, 2024

Contents

Chapter 1

Introduction

This document describes the format used for the Conlang (Constructed Language) inputs to the Constructed Language (Conlang) Audio Sampling program. Conlangs are input using a JSON ("ECMA-404," 2017) object, which will contain key information on the language, including its lexicon, morphology, spelling, pronunciation rules, and other information useful for generating speech. This format is designed to also serve as a general interchange format for conlangs that can be expanded and utilized for other tools, intending to be released as an open-source format.

In this document, a JSON Object containing a conlang will be called a *conlang object*.

Chapter 2

Capitalization

Key values in conlang objects will follow the same capitalization requirements as specified in "ECMA-404," 2017. Without any clear specification, the implementer should presume that all keys should be kept in all lowercase.

When an object of a key is a string that has limited valid values, the specification is for them to be in lowercase unless otherwise specified or recommended. In these cases, it is recommended that any implementation perform a case-insensitive comparison on these fields to minimize the chance of mismatching valid values.

Chapter 3

Top Level

The following are the keys that can be placed in the top level of a conlang object:

- **version**: Required. It must be 1.0 for this version of the document.
- **english_name**: Required. This specifies the name of the conlang in English or another natural language.
- **phonetic_characters**: Optional. This specifies the symbols being used to express the phonetics for the conlang. The valid values for this are **ipa**, **x-sampa**, and **sampa**. The default value if this field is not present is ipa.
- **native_name_phonetic**: Required. This specifies the name of the conlang phonetically. This will be expressed using the symbology specified in the **phonetic_characters** field.
- **native_name_english**: Optional. This species is the Romanized (Latinized) name of the conlang, which is how the conlang is expressed using characters in the Latin alphabet, possibly including diacritics.
- **preferred_voices**: Optional. This field contains a JSON object that has a map of speech-to-text applications and their preferred voices. Applications recognized at this

time are: “Polly,” “espeak-ng,” and “Azure.” Both the key and resulting field are capital sensitive due to the underlying programs.

- **preferred_language**: Optional. This specifies the preferred natural language for selecting phonemes for speaking this conlang. This should be specified using the XML:Lang specification from Froumentin, 2005.

- **derived**: Optional. This boolean specifies if the lexicon contains the results of processing the derived_word_list (below). If set to “true,” the lexicon contains these words, and any application should refrain from further attempts to derive these words. If set to false, these words may need to be derived before processing any conlang text. The default value is “false.”

- **declined**: Optional. This boolean specifies if the lexicon contains the results of processing the **affix_map** (below). If set to “true,” the lexicon contains all words’ declensions based upon the affix map. It is application-dependent if words are sensibly declined. For example, a declined lexicon may contain declensions for nouns that do not properly match their gender. The default value is “false.”

- **noun_gender_list**: Optional, Recommended. This array contains a list of strings defining the noun genders in the language. It is also recommended that two to three letters in this list be capitalized to be used as a short form for the gender in the **affix_map** (below).

- **part_of_speech_list**: Optional. This array contains a list of strings with abbreviations for the parts of speech that are contained within the lexicon. Applications can use this to help index or search the lexicon for key parts of speech.

- **phoneme_inventory**: Optional. This array contains a list of strings containing the allowed or expected phonemes in the conlang. These strings will be expressed using the symbology specified in the **phonetic_characters** field.

- **phonetic_inventory**: Optional. This object will contain the individual phonetic sounds (phones) and diphthongs found in the language. The keys of this object (all optional, but recommended) are "p_consonants" for the pulmonic consonants, "np_consonants" for the non-pulmonic consonants, "vowels," "v_diphthongs." Other keys can be added if needed. Each key will reference an array of strings containing either a single phone, represented by a single phonetic character, a character followed by a diacritic, or a pair of characters that may or may not be linked. These strings will be expressed using the symbology specified in the **phonetic_characters** field.

- **word_order**: Optional, Recommended. This string will contain the language's word order expressed using the letters "S" for subject, "V" for verb, and "O" for object. The language may take on any of the six possible arrangements for these.

- **adjective_position**: Optional, Recommended. This string indicates where adjectives are placed in this language in relationship to the noun that the adjective modifies. "Before" indicates that the adjective precedes the noun, as occurs in English, and "After" indicates that the adjective follows the noun it modifies, as occurs in French.

- **pre_post_position**: Optional, Recommended. This is used to specify if the language has prepositions or postpositions for its adposition. In languages with postpositions, the adposition comes after the noun phrase, for example, "a key with." In languages with prepositions, the adposition proceeds the noun phrase, for example, "with a key." The valid values are "preposition," and "postposition."

- **sound_map_list**: Optional, Recommended. This array of objects contains the objects described below that are used to map between the phonetic representation of the conlang and its Romanized or Latinized representation using the Latin alphabet. Without this entry, tools cannot translate between phonetic and Latin formats. This list is traversed in the order presented when generating the Latin alphabet version of

the word from the phonetic version. When generating the phonetic version of the word from the Latin alphabet version, it is traversed in the reverse of the order presented.

- **lexical_order_list**: Optional, Recommended. This array of strings is used to sort the words in the conlang into a lexical order after they have been converted into the Latin alphabet. If not provided, then the natural lexical order provided by the strings will be used for sorting.

- **affix_map**: Optional, Recommended. This object maps parts of speech, usually their abbreviations as listed in **part_of_speech_list**, to a list of objects described below that are used to decline root words of that part of speech.

- **derivational_affix_map**: Optional, Recommended. This object contains keys that are used to aid in creating derived words. The format of these objects is described below.

- **lexicon**: Required: This list of objects contains the lexicon, or dictionary, of the conlang. The format of these objects is described below.

- **derived_word_list**: Optional, Recommended. This array of strings contains words that will be derived to add to the lexicon. The format for these strings is borrowed from Vulgarlang (Cam from Vulgarlang, n.d.), and may be updated later. This format is described below.

- **metadata**: Optional, Recommended. This object contains an object where any program that edits the conlang object may add information regarding the history of the conlang and its previous processing. There is no exact format specified for this. Programs should not delete or alter metadata created by other programs but may add their own metadata or alter their metadata to update their content.

Chapter 4

Sound Map List Entries

Each entry in the sound_map_list array will be an object. This object has four keys in two groupings. These are:

- **pronunciation_regex**: Optional, Recommended, Required if **phoneme** is present. This string contains a generalized regular expression that matches a portion of a word's Romanized or Latinized version in the conlang with a specific phonetic representation. The matched text will then be replaced with the value from the **phoneme** below.
- **phoneme**: Optional, Recommended, Required if **pronunciation_regex** is present. This string contains the values that will be substituted for text matched by the **pronunciation_regex** when converting text from Romanized or Latinized to phonetic representation. This will be expressed using the symbology specified in the **phonetic_characters** field at the Top Level. The Perl standard of $n is used if group substitutions are included. If the application is written in a language or with a library such as Python that uses a different character, it is up to the application developer to change this character before utilizing this field.
- **spelling_regex**: Optional, Recommended, Required if **romanization** is present. This string contains a generalized regular expression used to match a portion of the

phonetic representation of a word in the conlang with a specific Romanization or Latinization. The matched text will then be replaced with the value from the **romanization** below. This will be expressed using the symbology specified in the **phonetic_characters** field at the Top Level.

- **romanization**: Optional, Recommended, Required if **spelling_regex** is present. This string contains the value that will be substituted for the text matched by the **spelling_regex** when converting text from phonetic representation. The Perl standard of $n is used if group substitutions are included. If the application is written in a language or with a library such as Python that uses a different character, it is up to the application developer to change this character before utilizing this field.

Generalized regular expressions are used here because they are powerful and can perform most or all of the substitutions required for mapping between the phonetic and the Romanized or Latinized representations of the conlang.

As described in Chapter 3, the Sound Map Entry List is traversed in forward order to convert from phonetic representation the Romanized or Latinized representation and is traversed in reverse order to convert to the phonetic representation. When designing the Sound Map List and the related Romanization or Latinization, or pronunciation schemes for the conlang, it is possible that these processes are not reciprocal. If this process and the Sound Map List cannot be fully reciprocal, then the conlang creator must prioritize which direction is most important for them. They will need to keep this priority in mind when utilizing this data and tools that use it since they may produce inconsistent or incorrect results when generating phonetic or Romanized or Latinized representations of words depending on the priority of the Sound Map List design.

It should be noted that many natural languages, including English, could not produce a reciprocal Sound Map List. Attempting to mimic the spelling rules of English or other natural languages will likely result in a Sound Map List that works best for mapping phonetic representation into Romanized or Latinized representation.

Chapter 5

Affix Map Entries

As described above, keys in the affix_map are parts of speech, and the values are lists of objects. Each object in the list will have a single key. The allowed values for this key are "particle," "prefix," "suffix," "replacement," and "pronunciation_add." The declension of the part of speech is dictated by the key. The value in each case will be a list of objects.

5.1 Particle

The use of "particle" is deprecated and not recommended. Its format will be reverse-engineered at a later point, replacing this deprecation warning. Until then, particles should be placed in the Lexicon instead.

5.2 Prefix and Suffix Objects

Each object in the list with a key of "prefix" or "suffix" will have a single key. This key will describe the type of declension represented by that entry. For example, a verb might have a declension of "Remote past Feminine" to indicate that combination of tense and gender. The value will be another object.

This object has eight keys. These are:

- **pronunciation_add**: Optional, Recommended, Required if **spelling_add** is present, Not allowed if **pronnounciation_regex** is present. This string contains the value that will be added to the beginning of a word for a prefix or the end of a word for a suffix in its phonetic representation. This will be expressed using the symbology specified in the **phonetic_characters** field at the Top Level.

- **spelling_add**: Optional, Recommended, Required if **pronunciation_add** is present, Not allowed if **spelling_regex** is present. This string contains the value that will be added to the beginning of a word for a prefix or the end of a word for a suffix in its Romanized or Latinized representation.

- **pronunciation_regex**: Optional, Recommended, Required if **t_pronunciation_add**, **f_pronunciation_add**, or **spelling_regex** are present, Not allowed if **pronunciation_add** is present. This string contains the generalized regular expression used to match against the phonetic representation of the word to determine which phonetic string to add to the beginning or end of that phonetic representation. If the phonetic representation of the word matches this pattern, then the **t_pronunciation_add** string should be added to the appropriate end of the phonetic representation. The **f_pronunciation_add** string should be added instead if it does not match. This pattern should be constructed to match the beginning or end of the string as appropriate for the type of affix being constructed.

- **t_pronunciation_add**: Optional, Recommended, Required if **pronouncation_regex** is present, Not allowed if **pronuncation_add** is present. The string to add to the beginning of the phonetic representation for a prefix, or the end of the phonetic representation for a suffix, when phonetic representation matches **pronunciation_regex**. This will be expressed using the symbology specified in the **phonetic_characters** field at the Top Level.

- **f_pronouncation_add**: Optional, Recommended, Required if

pronouncation_regex is present, Not allowed if **pronuncation_add** is present. The string to add to the beginning of the phonetic representation for a prefix, or the end of the phonetic representation for a suffix, when phonetic representation does not match **pronunciation_regex**. This will be expressed using the symbology specified in the **phonetic_characters** field at the Top Level.

- **spelling_regex**: Optional, Recommended, Required if **t_spelling_add**, **f_spelling_add**, or **pronunciation_regex** are present, Not allowed if **spelling_add** is present. This string contains the generalized regular expression used to match against the Romanized or Latinized representation of the word to determine which Romanized or Latinized string to add to the beginning or end of that Romanized or Latinized representation. If the Romanized or Latinized representation of the word matches this pattern, then the **t_spelling_add** string should be added to the appropriate end of the phonetic representation. The **f_spelling_add** string should be added if it does not match. This pattern should be constructed to match the beginning or end of the string as appropriate for the type of affix being constructed.

- **t_spelling_add**: Optional, Recommended, Required if **spelling_regex** is present, Not allowed if **spelling_add** is present. The string to add to the beginning of the Romanized or Latinized representation for a prefix, or the end of the phonetic representation for a suffix, when the Romanized or Latinized representation matches **spelling_regex**.

- **f_spelling_add**: Optional, Recommended, Required if **spelling_regex** is present, Not allowed if **spelling_add** is present. The string to add to the beginning of the phonetic representation for a prefix, or the end of the Romanized or Latinized representation for a suffix, when the Romanized or Latinized representation does not match **spelling_regex**.

5.3 Replacement Objects

Each object in the list with a key of "replacement" will have a single key. This key will describe the type of declension represented by that entry. For example, a verb might have a declension of "Perfect" to indicate that aspect. The value will be another object.
This object will have four keys.

- **pronunciation_regex**: Optional, Recommended, Required if **pronunciation_replacement** is present. This is the generalized regular expression used to match the phonetic representation of the word to perform the substitution when declining the word in this instance. This should include one or more capture groups.

- **pronunciation_replacement**: Optional, Recommended, Required if **pronunciation_regex** is present. This is the pattern that will be used to replace the pattern matched by **pronunciation_regex** in the phonetic representation of the word when a match occurs. Replacement groups are represented using the Perl standard of $n. If the application is written in a language or with a library such as Python that uses a different character, it is up to the application developer to change this character before utilizing this field. This will be expressed using the symbology specified in the **phonetic_characters** field at the Top Level.

- **spelling_regex**: Optional, Recommended, Required if **spelling_replacement** is present. This is the generalized regular expression used to match the Romanized or Latinized representation of the word to perform the substitution when declining the word in this instance. This should include one or more capture groups.

- **spelling_replacement**: Optional, Recommended, Required if **spelling_regex** is present. This is the pattern that will be used to replace the pattern matched by **spelling_regex** in the Romanized or Latinized representation of the word when a

match occurs. Replacement groups are represented using the Perl standard of $n. If the application is written in a language or with a library such as Python that uses a different character, it is up to the application developer to change this character before utilizing this field.

Chapter 6

Derivational Affix Entries

As described above, the keys in the derivational_affix_map are used when deriving words. The entries are objects. These objects have nine keys.

- **type**: Required. This string has two valid values: "PREFIX" or "SUFFIX". This indicates if the affix placed on the root word when deriving the new word is a prefix or suffix.
- **pronunciation_add**: Optional, Recommended, Not allowed if **pronunciation_regex** or **spelling_regex** is present. This is the string to be prepended or appended to the phonetic representation of the root word to create the new derived word's phonetic representation. This will be expressed using the symbology specified in the **phonetic_characters** field at the Top Level.
- **spelling_add**: Optional, Recommended, Not allowed if **spelling_regex** or **pronunciation_regex** is present. This is the string to be prepended or appended to the Romanized or Latinized representation of the root word to create the new derived word's Romanized or Latinized representation.
- **pronunciation_regex**: Optional, Recommended, Required if **t_pronunciation_add** or **f_pronunciation_add** are present, Not allowed if

pronunciation_add or **spelling_add** is present. This is the generalized regular expression used to match on the phonetic representation of the root word to determine which phonetic string to prepend or append to that root word when creating the derived word.

- **t_pronunciation_add**: Optional, Recommended, Required if **pronunciation_regex** is present, Not allowed if **pronunciation_add** or **spelling_add** is present. This is the string to be prepended or appended to the phonetic representation of the root word to create the new derived word's phonetic representation if the root word matches the pattern in **pronouncation_regex**. This will be expressed using the symbology specified in the **phonetic_characters** field at the Top Level.

- **f_pronunciation_add**: Optional, Recommended, Required if **pronunciation_regex** is present, Not allowed if **pronunciation_add** or **spelling_add** is present. This is the string to be prepended or appended to the phonetic representation of the root word to create the new derived word's phonetic representation if the root word does not match the pattern in **pronouncation_regex**. This will be expressed using the symbology specified in the **phonetic_characters** field at the Top Level.

- **spelling_regex**: Optional, Recommended, Required if **t_spelling_add** or **f_spelling_add** are present, Not allowed if **pronuncation_add** or **spelling_add** is present. This is the generalized regular expression used to match on the Romanized or Latinized representation of the root word to determine which Romanized or Latinized string to prepend or append to that root word when creating the derived word.

- **t_spelling_add**: Optional, Recommended, Required if **spelling_regex** is present, Not allowed if **pronunciation_add** or **spelling_add** is present. This is the string

to be prepended or appended to the Romanized or Latinized representation of the root word to create the new word's Romanized or Latinized representation if the root word matches the pattern in **spelling_regex**.

- **f_spelling_add**: Optional, Recommended, Required if **spelling_regex** is present, Not allowed if **pronunciation_add** or **spelling_add** is present. This is the string to be prepended or appended to the Romanized or Latinized representation of the root word to create the new word's Romanized or Latinized representation if the root word does not match the pattern in **spelling_regex**.

Chapter 7

Lexicon Entries

As described above, the lexicon will be a list of objects. Each object will have the following keys:

- **phonetic**: Optional. Required if **spelled** is not present. This string contains the phonetic representation of the word. This will be expressed using the symbology specified in the **phonetic_characters** field at the Top Level.
- **spelled**: Optional. Required if **phonetic** is not present. This string contains the Romanized or Latinized representation of the word.
- **english**: Required. This string contains the English or other natural language equivalent to the word or its definition in English or another natural language.
- **part_of_speech**: Required. This string contains the part of speech for this word. This can, and should, be one of the abbreviations found in the **part_of_speech_list** field at the Top Level.
- **declensions**: Required. This array of strings contains the declensions used when declining this word from its root form. If the word is not declined, that is, if it is the root word, this array must contain a single entry, the string "root."

- **derived_word**: Required. This boolean is set to true if this word was created by deriving a root word using words in the **derived_word_list** with the aid of the affixes in the **derivation_affix_map**.
- **declined_word**: Required. This boolean is set to true if the word was created by declining a root word using the rules in the **affix_map**.
- **metadata**: Optionall, Recommended. This object contains an object where any program that edits the conlang object may add information regarding the word's history and its source. There is no exact format specified for this. Programs should not delete or alter metadata created by other programs but may add their own metadata or alter their metadata to update their content.

Bibliography

Cam from Vulgarlang. (n.d.). *Vulgarlang, A Fantasy Language Generator*. https://www.vulgarlang.com/

Froumentin, M. (2005, November). *The W3C Speech Interface Framework Media Types: Application/voicexml+xml, application/ssml+xml, application/srgs, application/srgs+xml, application/ccxml+xml, and application/pls+xml*. Retrieved October 24, 2023, from https://www.ietf.org/rfc/rfc4267.txt

The JSON Data Interchange Syntax. (2017, December). https://ecma-international.org/publications-and-standards/standards/ecma-404/