Network Working Group
Request for Comments: 4647
BCP: 47
Obsoletes: 3066
Category: Best Current Practice

A. Phillips, Ed.
Yahoo! Inc.
M. Davis, Ed.
Google

September 2006

Matching of Language Tags

Status of This Memo

Copyright Notice

Abstract

This document describes a syntax, called a "language-range", for specifying items in a user's list of language preferences. It also describes different mechanisms for comparing and matching these to language tags. Two kinds of matching mechanisms, filtering and lookup, are defined. Filtering produces a (potentially empty) set of language tags, whereas lookup produces a single language tag. Possible applications include language negotiation or content selection. This document, in combination with RFC 4646, replaces RFC 3066, which replaced RFC 1766.

Table of Contents

1. Introduction

Human beings on our planet have, past and present, used a number of languages. There are many reasons why one would want to identify the language used when presenting or requesting information.

Applications, protocols, or specifications that use language identifiers, such as the language tags defined in [RFC4646], sometimes need to match language tags to a user's language preferences.

This document defines a syntax (called a language range (Section 2)) for specifying items in the user's list of language preferences (called a language priority list (Section 2.3)), as well as several schemes for selecting or filtering sets of language tags by comparing the language tags to the user's preferences. Applications, protocols, or specifications will have varying needs and requirements that affect the choice of a suitable matching scheme.

This document describes how to indicate a user's preferences using language ranges, three schemes for matching these ranges to a set of language tags, and the various practical considerations that apply to implementing and using these schemes.

This document, in combination with [RFC4646], replaces [RFC3066], which replaced [RFC1766].

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119].

2. The Language Range

Language tags [RFC4646] are used to help identify languages, whether spoken, written, signed, or otherwise signaled, for the purpose of communication. Applications, protocols, or specifications that use language tags are often faced with the problem of identifying sets of content that share certain language attributes. For example, HTTP/1.1 [RFC2616] describes one such mechanism in its discussion of the Accept-Language header (Section 14.4), which is used when selecting content from servers based on the language of that content.

It is, thus, useful to have a mechanism for identifying sets of language tags that share specific attributes. This allows users to select or filter the language tags based on specific requirements. Such an identifier is called a "language range".

There are different types of language range, whose specific attributes vary according to their application. Language ranges are similar to language tags: they consist of a sequence of subtags separated by hyphens. In a language range, each subtag MUST either be a sequence of ASCII alphanumeric characters or the single character '*' (%x2A, ASTERISK). The character '*' is a "wildcard" that matches any sequence of subtags. The meaning and uses of wildcards vary according to the type of language range.

Language tags and thus language ranges are to be treated as case-insensitive: there exist conventions for the capitalization of some of the subtags, but these MUST NOT be taken to carry meaning. Matching of language tags to language ranges MUST be done in a case-insensitive manner.

2.1. Basic Language Range

A "basic language range" has the same syntax as an [RFC3066] language tag or is the single character "*". The basic language range was originally described by HTTP/1.1 [RFC2616] and later [RFC3066]. It is defined by the following ABNF [RFC4234]:

```
language-range   = (1*8ALPHA *("-" 1*8alphanum)) / "*"
alphanum         = ALPHA / DIGIT
```

A basic language range differs from the language tags defined in [RFC4646] only in that there is no requirement that it be "well-formed" or be validated against the IANA Language Subtag Registry. Such ill-formed ranges will probably not match anything. Note that the ABNF [RFC4234] in [RFC2616] is incorrect, since it disallows the use of digits anywhere in the 'language-range' (see [RFC2616errata]).

2.2. Extended Language Range

Occasionally, users will wish to select a set of language tags based on the presence of specific subtags. An "extended language range" describes a user's language preference as an ordered sequence of subtags. For example, a user might wish to select all language tags that contain the region subtag 'CH' (Switzerland). Extended language ranges are useful for specifying a particular sequence of subtags that appear in the set of matching tags without having to specify all of the intervening subtags.

An extended language range can be represented by the following ABNF:

```
extended-language-range = (1*8ALPHA / "*")
                          *("-" (1*8alphanum / "*"))
```

The wildcard subtag '*' can occur in any position in the extended language range, where it matches any sequence of subtags that might occur in that position in a language tag. However, wildcards outside the first position are ignored by Extended Filtering (see Section 3.2.2). The use or absence of one or more wildcards cannot be taken to imply that a certain number of subtags will appear in the matching set of language tags.

2.3. The Language Priority List

A user's language preferences will often need to specify more than one language range, and thus users often need to specify a prioritized list of language ranges in order to best reflect their language preferences. This is especially true for speakers of minority languages. A speaker of Breton in France, for example, can specify "br" followed by "fr", meaning that if Breton is available, it is preferred, but otherwise French is the best alternative. It can get more complex: a different user might want to fall back from Skolt Sami to Northern Sami to Finnish.

A "language priority list" is a prioritized or weighted list of language ranges. One well-known example of such a list is the "Accept-Language" header defined in RFC 2616 [RFC2616] (see Section 14.4) and RFC 3282 [RFC3282].

The various matching operations described in this document include considerations for using a language priority list. This document does not define the syntax for a language priority list; defining such a syntax is the responsibility of the protocol, application, or specification that uses it. When given as examples in this document, language priority lists will be shown as a quoted sequence of ranges separated by commas, like this: "en, fr, zh-Hant" (which is read "English before French before Chinese as written in the Traditional script").

A simple list of ranges is considered to be in descending order of priority. Other language priority lists provide "quality weights" for the language ranges in order to specify the relative priority of the user's language preferences. An example of this is the use of "q" values in the syntax of the "Accept-Language" header (defined in [RFC2616], Section 14.4, and [RFC3282]).

3. Types of Matching

Matching language ranges to language tags can be done in many different ways. This section describes three such matching schemes, as well as the considerations for choosing between them. Protocols and specifications requiring conformance to this specification MUST clearly indicate the particular mechanism used in selecting or matching language tags.

There are two types of matching scheme in this document. A matching scheme that produces zero or more matching language tags is called "filtering". A matching scheme that produces exactly one match for a given request is called "lookup".

3.1. Choosing a Matching Scheme

Applications, protocols, and specifications are faced with the decision of what type of matching to use. Sometimes, different styles of matching are suited to different kinds of processing within a particular application or protocol.

This document describes three matching schemes:

1. Basic Filtering (Section 3.3.1) matches a language priority list consisting of basic language ranges (Section 2.1) to sets of language tags.

2. Extended Filtering (Section 3.3.2) matches a language priority list consisting of extended language ranges (Section 2.2) to sets of language tags.

3. Lookup (Section 3.4) matches a language priority list consisting of basic language ranges to sets of language tags to find the one exact language tag that best matches the range.

Filtering can be used to produce a set of results (such as a collection of documents) by comparing the user's preferences to a set of language tags. For example, when performing a search, filtering can be used to limit the results to items tagged as being in the French language. Filtering can also be used when deciding whether to perform a language-sensitive process on some content. For example, a process might cause paragraphs whose language tag matched the language range "nl" (Dutch) to be displayed in italics within a document.

Lookup produces the single result that best matches the user's preferences from the list of available tags, so it is useful in cases in which a single item is required (and for which only a single item

can be returned). For example, if a process were to insert a human-readable error message into a protocol header, it might select the text based on the user's language priority list. Since the process can return only one item, it is forced to choose a single item and it has to return some item, even if none of the content's language tags match the language priority list supplied by the user.

3.2. Implementation Considerations

Language tag matching is a tool, and does not by itself specify a complete procedure for the use of language tags. Such procedures are intimately tied to the application protocol in which they occur. When specifying a protocol operation using matching, the protocol MUST specify:

- Which type(s) of language tag matching it uses
- Whether the operation returns a single result (lookup) or a possibly empty set of results (filtering)
- For lookup, what the default item is (or the sequence of operations or configuration information used to determine the default) when no matching tag is found. For instance, a protocol might define the result as failure of the operation, an empty value, returning some protocol defined or implementation defined default, or returning i-default [RFC2277].

Applications, protocols, and specifications are not required to validate or understand any of the semantics of the language tags or ranges or of the subtags in them, nor do they require access to the IANA Language Subtag Registry (see Section 3 in [RFC4646]). This simplifies implementation.

However, designers of applications, protocols, or specifications are encouraged to use the information from the IANA Language Subtag Registry to support canonicalizing language tags and ranges in order to map grandfathered and obsolete tags or subtags into modern equivalents.

Applications, protocols, or specifications that canonicalize ranges MUST either perform matching operations with both the canonical and original (unmodified) form of the range or MUST also canonicalize each tag for the purposes of comparison.

Note that canonicalizing language ranges makes certain operations impossible. For example, an implementation that canonicalizes the language range "art-lojban" (artificial language, lojban variant) to use the more modern "jbo" (Lojban) cannot be used to select just the items with the older tag.

Applications, protocols, or specifications that use basic ranges might sometimes receive extended language ranges instead. An application, protocol, or specification MUST choose to a) map extended language ranges to basic ranges using the algorithm below, b) reject any extended language ranges in the language priority list that are not valid basic language ranges, or c) treat each extended language range as if it were a basic language range, which will have the same result as ignoring them, since these ranges will not match any valid language tags.

An extended language range is mapped to a basic language range as follows: if the first subtag is a '*' then the entire range is treated as "*", otherwise each wildcard subtag is removed. For example, the extended language range "en-*-US" maps to "en-US" (English, United States).

Applications, protocols, or specifications, in addressing their particular requirements, can offer pre-processing or configuration options. For example, an implementation could allow a user to associate or map a particular language range to a different value. Such a user might wish to associate the language range subtags 'nn' (Nynorsk Norwegian) and 'nb' (Bokmal Norwegian) with the more general subtag 'no' (Norwegian). Or perhaps a user would want to associate requests for the range "zh-Hans" (Chinese as written in the Simplified script) with content bearing the language tag "zh-CN" (Chinese as used in China, where the Simplified script is predominant). Documentation on how the ranges or tags are altered, prioritized, or compared in the subsequent match in such an implementation will assist users in making these types of configuration choices.

3.3. Filtering

Filtering is used to select the set of language tags that matches a given language priority list. It is called "filtering" because this set might contain no items at all or it might return an arbitrarily large number of matching items: as many items as match the language priority list, thus "filtering out" the non-matching items.

In filtering, each language range represents the least specific language tag (that is, the language tag with fewest number of subtags) that is an acceptable match. All of the language tags in

the matching set of tags will have an equal or greater number of subtags than the language range. Every non-wildcard subtag in the language range will appear in every one of the matching language tags. For example, if the language priority list consists of the range "de-CH" (German as used in Switzerland), one might see tags such as "de-CH-1996" (German as used in Switzerland, orthography of 1996) but one will never see a tag such as "de" (because the 'CH' subtag is missing).

If the language priority list (see Section 2.3) contains more than one range, the content returned is typically ordered in descending level of preference, but it MAY be unordered, according to the needs of the application or protocol.

Some examples of applications where filtering might be appropriate include:

- Applying a style to sections of a document in a particular set of languages.
- Displaying the set of documents containing a particular set of keywords written in a specific set of languages.
- Selecting all email items written in a specific set of languages.
- Selecting audio files spoken in a particular language.

Filtering seems to imply that there is a semantic relationship between language tags that share the same prefix. While this is often the case, it is not always true: the language tags that match a specific language range do not necessarily represent mutually intelligible languages.

3.3.1. Basic Filtering

Basic filtering compares basic language ranges to language tags. Each basic language range in the language priority list is considered in turn, according to priority. A language range matches a particular language tag if, in a case-insensitive comparison, it exactly equals the tag, or if it exactly equals a prefix of the tag such that the first character following the prefix is "-". For example, the language-range "de-de" (German as used in Germany) matches the language tag "de-DE-1996" (German as used in Germany, orthography of 1996), but not the language tags "de-Deva" (German as written in the Devanagari script) or "de-Latn-DE" (German, Latin script, as used in Germany).

The special range "*" in a language priority list matches any tag. A protocol that uses language ranges MAY specify additional rules about the semantics of "*"; for instance, HTTP/1.1 [RFC2616] specifies that the range "*" matches only languages not matched by any other range within an "Accept-Language" header.

Basic filtering is identical to the type of matching described in [RFC3066], Section 2.5 (Language-range).

3.3.2. Extended Filtering

Extended filtering compares extended language ranges to language tags. Each extended language range in the language priority list is considered in turn, according to priority. A language range matches a particular language tag if each respective list of subtags matches. To determine a match:

1. Split both the extended language range and the language tag being compared into a list of subtags by dividing on the hyphen (%x2D) character. Two subtags match if either they are the same when compared case-insensitively or the language range's subtag is the wildcard '*'.

2. Begin with the first subtag in each list. If the first subtag in the range does not match the first subtag in the tag, the overall match fails. Otherwise, move to the next subtag in both the range and the tag.

3. While there are more subtags left in the language range's list:

   A. If the subtag currently being examined in the range is the wildcard ('*'), move to the next subtag in the range and continue with the loop.

   B. Else, if there are no more subtags in the language tag's list, the match fails.

   C. Else, if the current subtag in the range's list matches the current subtag in the language tag's list, move to the next subtag in both lists and continue with the loop.

   D. Else, if the language tag's subtag is a "singleton" (a single letter or digit, which includes the private-use subtag 'x') the match fails.

   E. Else, move to the next subtag in the language tag's list and continue with the loop.

4. When the language range's list has no more subtags, the match succeeds.

Subtags not specified, including those at the end of the language range, are thus treated as if assigned the wildcard value '*'. Much like basic filtering, extended filtering selects content with arbitrarily long tags that share the same initial subtags as the language range. In addition, extended filtering selects language tags that contain any intermediate subtags not specified in the language range. For example, the extended language range "de-*-DE" (or its synonym "de-DE") matches all of the following tags:

de-DE (German, as used in Germany)

de-de (German, as used in Germany)

de-Latn-DE (Latin script)

de-Latf-DE (Fraktur variant of Latin script)

de-DE-x-goethe (private-use subtag)

de-Latn-DE-1996 (orthography of 1996)

de-Deva-DE (Devanagari script)

The same range does not match any of the following tags for the reasons shown:

de (missing 'DE')

de-x-DE (singleton 'x' occurs before 'DE')

de-Deva ('Deva' not equal to 'DE')

Note: [RFC4646] defines each type of subtag (language, script, region, and so forth) according to position, size, and content. This means that subtags in a language range can only match specific types of subtags in a language tag. For example, a subtag such as 'Latn' is always a script subtag (unless it follows a singleton) while a subtag such as 'nedis' can only match the equivalent variant subtag. Two-letter subtags in the initial position have a different type (language) than two-letter subtags in later positions (region). This is the reason why a wildcard in the extended language range is significant in the first position but is ignored in all other positions.

3.4. Lookup

Lookup is used to select the single language tag that best matches the language priority list for a given request. When performing lookup, each language range in the language priority list is considered in turn, according to priority. By contrast with filtering, each language range represents the most specific tag that is an acceptable match. The first matching tag found, according to the user's priority, is considered the closest match and is the item returned. For example, if the language range is "de-ch", a lookup operation can produce content with the tags "de" or "de-CH" but never content with the tag "de-CH-1996". If no language tag matches the request, the "default" value is returned.

For example, if an application inserts some dynamic content into a document, returning an empty string if there is no exact match is not an option. Instead, the application "falls back" until it finds a matching language tag associated with a suitable piece of content to insert. Some applications of lookup include:

- Selection of a template containing the text for an automated email response.
- Selection of an item containing some text for inclusion in a particular Web page.
- Selection of a string of text for inclusion in an error log.
- Selection of an audio file to play as a prompt in a phone system.

In the lookup scheme, the language range is progressively truncated from the end until a matching language tag is located. Single letter or digit subtags (including both the letter 'x', which introduces private-use sequences, and the subtags that introduce extensions) are removed at the same time as their closest trailing subtag. For example, starting with the range "zh-Hant-CN-x-private1-private2" (Chinese, Traditional script, China, two private-use tags) the lookup progressively searches for content as shown below:

Example of a Lookup Fallback Pattern

Range to match: zh-Hant-CN-x-private1-private2
1. zh-Hant-CN-x-private1-private2
2. zh-Hant-CN-x-private1
3. zh-Hant-CN
4. zh-Hant
5. zh
6. (default)

This fallback behavior allows some flexibility in finding a match. Without fallback, the default content would be returned immediately if exactly matching content is unavailable. With fallback, a result more closely matching the user request can be provided.

Extensions and unrecognized private-use subtags might be unrelated to a particular application of lookup. Since these subtags come at the end of the subtag sequence, they are removed first during the fallback process and usually pose no barrier to interoperability. However, an implementation MAY remove these from ranges prior to performing the lookup (provided the implementation also removes them from the tags being compared). Such modification is internal to the implementation and applications, protocols, or specifications SHOULD NOT remove or modify subtags in content that they return or forward, because this removes information that can be used elsewhere.

The special language range "*" matches any language tag. In the lookup scheme, this range does not convey enough information by itself to determine which language tag is most appropriate, since it matches everything. If the language range "*" is followed by other language ranges, it is skipped. If the language range "*" is the only one in the language priority list or if no other language range follows, the default value is computed and returned.

In some cases, the language priority list can contain one or more extended language ranges (as, for example, when the same language priority list is used as input for both lookup and filtering operations). Wildcard values in an extended language range normally match any value that can occur in that position in a language tag. Since only one item can be returned for any given lookup request, wildcards in a language range have to be processed in a consistent manner or the same request will produce widely varying results. Applications, protocols, or specifications that accept extended language ranges MUST define which item is returned when more than one item matches the extended language range.

For example, an implementation could map the extended language ranges to basic ranges. Another possibility would be for an implementation to return the matching tag that is first in ASCII-order. If the language range were "*-CH" (’CH’ represents Switzerland) and the set of tags included "de-CH" (German as used in Switzerland), "fr-CH" (French, Switzerland), and "it-CH" (Italian, Switzerland), then the tag "de-CH" would be returned.

3.4.1. Default Values

Each application, protocol, or specification that uses lookup MUST define the defaulting behavior when no tag matches the language priority list. What this action consists of strongly depends on how lookup is being applied. Some examples of defaulting behavior include:

- return an item with no language tag or an item of a non-linguistic nature, such as an image or sound
- return a null string as the language tag value, in cases where the protocol permits the empty value (see, for example, "xml:lang" in [XML10])
- return a particular language tag designated for the operation
- return the language tag "i-default" (see [RFC2277])
- return an error condition or error message
- return a list of available languages for the user to select from

When performing lookup using a language priority list, the progressive search MUST process each language range in the list before seeking or calculating the default.

The default value MAY be calculated or include additional searching or matching. Applications, protocols, or specifications can specify different ways in which users can specify or override the defaults.

One common way to provide for a default is to allow a specific language range to be set as the default for a specific type of request. If this approach is chosen, this language range MUST be treated as if it were appended to the end of the language priority list as a whole, rather than after each item in the language priority list. The application, protocol, or specification MUST also define the defaulting behavior if that search fails to find a matching tag or item.

For example, if a particular user's language priority list is "fr-FR, zh-Hant" (French as used in France followed by Chinese as written in the Traditional script) and the program doing the matching had a default language range of "ja-JP" (Japanese as used in Japan), then the program searches as follows:

1. fr-FR
2. fr
3. zh-Hant // next language
4. zh
5. ja-JP // now searching for the default content
6. ja
7. (implementation defined default)

4. Other Considerations

When working with language ranges and matching schemes, there are some additional points that can influence the choice of either.

4.1. Choosing Language Ranges

Users indicate their language preferences via the choice of a language range or the list of language ranges in a language priority list. The type of matching affects what the best choice is for a user.

Most matching schemes make no attempt to process the semantic meaning of the subtags. The language range is compared, in a case-insensitive manner, to each language tag being matched, using basic string processing. Users SHOULD select language ranges that are well-formed, valid language tags according to [RFC4646] (substituting wildcards as appropriate in extended language ranges).

Applications are encouraged to canonicalize language tags and ranges by using the Preferred-Value from the IANA Language Subtag Registry for tags or subtags that have been deprecated. If the user is working with content that might use the older form, the user might want to include both the new and old forms in a language priority list. For example, the tag "art-lojban" is deprecated. The subtag ’jbo’ is supposed to be used instead, so the user might use it to form the language range. Or the user might include both in a language priority list: "jbo, art-lojban".

Users SHOULD avoid subtags that add no distinguishing value to a language range. When filtering, the fewer the number of subtags that appear in the language range, the more content the range will probably match, while in lookup unnecessary subtags can cause "better", more-specific content to be skipped in favor of less specific content. For example, the range "de-Latn-DE" returns content tagged "de" instead of content tagged "de-DE", even though the latter is probably a better match.

Whether a subtag adds distinguishing value can depend on the context of the request. For example, a user who reads both Simplified and Traditional Chinese, but who prefers Simplified, might use the range "zh" for filtering (matching all items that user can read) but "zh-Hans" for lookup (making sure that user gets the preferred form if it's available, but the fallback to "zh" will still work). On the other hand, content in this case ought to be labeled as "zh-Hans" (or "zh-Hant" if that applies) for filtering, while for lookup, if there is either "zh-Hans" content or "zh-Hant" content, one of them (the one considered 'default') also ought to be made available with the simple "zh". Note that the user can create a language priority list "zh-Hans, zh" that delivers the best possible results for both schemes. If the user cannot be sure which scheme is being used (or if more than one might be applied to a given request), the user SHOULD specify the most specific (largest number of subtags) range first and then supply shorter prefixes later in the list to ensure that filtering returns a complete set of tags.

Many languages are written predominantly in a single script. This is usually recorded in the Suppress-Script field in that language subtag's registry entry. For these languages, script subtags SHOULD NOT be used to form a language range. Thus, the language range "en-Latn" is inappropriate in most cases (because the vast majority of English documents are written in the Latin script and thus the 'en' language subtag has a Suppress-Script field for 'Latn' in the registry).

When working with tags and ranges, note that extensions and most private-use subtags are orthogonal to language tag matching, in that they specify additional attributes of the text not related to the goals of most matching schemes. Users SHOULD avoid using these subtags in language ranges, since they interfere with the selection of available content. When used in language tags (as opposed to ranges), these subtags normally do not interfere with filtering (Section 3), since they appear at the end of the tag and will match all prefixes. Lookup (Section 3.4) implementations are advised to ignore unrecognized private-use and extension subtags when performing language tag fallback.

4.2. Meaning of Language Tags and Ranges

Selecting language tags using language ranges requires some understanding by users of what they are selecting. The meanings of the various subtags in a language range are identical to their meanings in a language tag (see Section 4.2 in [RFC4646]), with the addition that the wildcard "*" represents any matching sequence of values.

4.3. Considerations for Private-Use Subtags

Private agreement is necessary between the parties that intend to use or exchange language tags that contain private-use subtags. Great caution SHOULD be used in employing private-use subtags in content or protocols intended for general use. Private-use subtags are simply useless for information exchange without prior arrangement.

The value and semantic meaning of private-use tags and of the subtags used within such a language tag are not defined. Matching private-use tags using language ranges or extended language ranges can result in unpredictable content being returned.

4.4. Length Considerations for Language Ranges

Language ranges are very similar to language tags in terms of content and usage. The same types of restrictions on length that can be applied to language tags can also be applied to language ranges. See [RFC4646] Section 4.3 (Length Considerations).

5. Security Considerations

Language ranges used in content negotiation might be used to infer the nationality of the sender, and thus identify potential targets for surveillance. In addition, unique or highly unusual language ranges or combinations of language ranges might be used to track a specific individual’s activities.

This is a special case of the general problem that anything you send is visible to the receiving party. It is useful to be aware that such concerns can exist in some cases.

The evaluation of the exact magnitude of the threat, and any possible countermeasures, is left to each application or protocol.

6. Character Set Considerations

Language tags permit only the characters A-Z, a-z, 0-9, and HYPHEN-MINUS (%x2D). Language ranges also use the character ASTERISK (%x2A). These characters are present in most character sets, so presentation or exchange of language tags or ranges should not be constrained by character set issues.

7. References

7.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC2277] Alvestrand, H., "IETF Policy on Character Sets and Languages", BCP 18, RFC 2277, January 1998.

[RFC4234] Crocker, D., Ed. and P. Overell, "Augmented BNF for Syntax Specifications: ABNF", RFC 4234, October 2005.

[RFC4646] Phillips, A., Ed., and M. Davis, Ed., "Tags for Identifying Languages", BCP 47, RFC 4646, September 2006.

7.2. Informative References

[RFC1766] Alvestrand, H., "Tags for the Identification of Languages", RFC 1766, March 1995.

[RFC2616] Fielding, R., Gettys, J., Mogul, J., Frystyk, H., Masinter, L., Leach, P., and T. Berners-Lee, "Hypertext Transfer Protocol -- HTTP/1.1", RFC 2616, June 1999.

[RFC2616errata] IETF, "HTTP/1.1 Specification Errata", October 2004, <http://purl.org/NET/http-errata>.

[RFC3066] Alvestrand, H., "Tags for the Identification of Languages", BCP 47, RFC 3066, January 2001.

[RFC3282] Alvestrand, H., "Content Language Headers", RFC 3282, May 2002.

[XML10] Bray, T., Paoli, J., Sperberg-McQueen, C., Maler, E., and F. Yergeau, "Extensible Markup Language (XML) 1.0 (Third Edition)", World Wide Web Consortium Recommendation, February 2004, <http://www.w3.org/TR/REC-xml>.

Appendix A. Acknowledgements

Any list of contributors is bound to be incomplete; please regard the following as only a selection from the group of people who have contributed to make this document what it is today.

The contributors to [RFC1766] and [RFC3066], each of which was a precursor to this document, contributed greatly to the development of language tag matching, and, in particular, the basic language range and the basic matching scheme. This document was originally part of [RFC4646], but was split off before that document's completion. Thus, directly or indirectly, those acknowledged in [RFC4646] also had a hand in the development of this document, and work done prior to the split is acknowledged in that document.

The following people (in alphabetical order by family name) contributed to this document:

Harald Alvestrand, Stephane Bortzmeyer, Jeremy Carroll, Peter Constable, John Cowan, Mark Crispin, Martin Duerst, Frank Ellermann, Doug Ewell, Debbie Garside, Marion Gunn, Jon Hanna, Kent Karlsson, Erkki Kolehmainen, Jukka Korpela, Ira McDonald, M. Patton, Randy Presuhn, Eric van der Poel, Markus Scherer, Misha Wolf, and many, many others.

Very special thanks must go to Harald Tveit Alvestrand, who originated RFCs 1766 and 3066, and without whom this document would not have been possible.

Authors' Addresses

Addison Phillips (Editor)
Yahoo! Inc.

EMail: addison@inter-locale.com

Mark Davis (Editor)
Google

EMail: mark.davis@macchiato.com or mark.davis@google.com

Full Copyright Statement

Intellectual Property

Acknowledgement

Funding for the RFC Editor function is provided by the IETF Administrative Support Activity (IASA).