AI Preferences
Internet-Draft
Updates: 9309 (if approved)
Intended status: Standards Track
Expires: 21 August 2025

M. Thomson
Mozilla

17 February 2025

Short Usage Preference Strings for Automated Processing
draft-thomson-aipref-sup-00

Abstract

Content creators and other stakeholders might wish to signal their preferences about how their content might be consumed by automated systems. This document defines a very simple format for expressions.

This document updates RFC 9309 to define one means of conveyance. An HTTP header field is also defined.

About This Document

This note is to be removed before publishing as an RFC.

The latest revision of this draft can be found at https://martinthomson.github.io/sup-ai/draft-thomson-aipref-sup.html. Status information for this document may be found at https://datatracker.ietf.org/doc/draft-thomson-aipref-sup/.

Discussion of this document takes place on the AI Preferences Working Group mailing list (mailto:ai-control@ietf.org), which is archived at https://mailarchive.ietf.org/arch/browse/ai-control/. Subscribe at https://www.ietf.org/mailman/listinfo/ai-control/.

Source for this draft and an issue tracker can be found at https://github.com/martinthomson/sup-ai.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

The automated consumption of content by crawlers and other machines has increased significantly in recent years. This is partly due to the training of machine-learning models.

Content creators and other stakeholders, such as distributors, might wish to have a say in what types of usage by automatons is acceptable. At the same time, the operator of an automated system might be happy to follow any guidance given.

In the absence of a clear means of indicating preferences, there is no way for preferences to conveyed. This document seeks to address this shortcoming.

The format of preferences is a simple string that looks something like this:

ai=n,search=y

This format seeks to be:

* Simple to understand and create

* Straightforward to consume

* Easily conveyed in multiple ways

The document defines extensions to "robots.txt" [ROBOTS] and HTTP [HTTP] for making preferences available to automated processing systems.

1.1. Applicability

These expressions do not consider the process of acquiring content. They only express preferences regarding how the content is ultimately used. In particular, this includes both the training of machine learning models and the use of those models.

This is not intended to replace copyright licensing or other means of conveying similar information. Rather, it is intended to provide clear information that can be used by automatons to quickly identify whether content is available or not for processing.

Preference expressions do not apply where usage is explicitly allowed or disallowed in law or when licensing has been pre-arranged.

This document assumes that automatons have a default policy that applies when no preferences have been expressed. How those defaults are determined is not specified.

1.2. Conventions and Definitions

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

2. Examples

The following string opts out of all forms of automated processing for the associated content:

```
tdm=n
```

On its own, this string does not define which content is affected.

The following shows how a preference to allow general text and data mining, except for artificial intelligence applications, is applied to content that is delivered in an HTTP response; see Section 6.2.

```
404 Not Found
Date: Mon, 17 Feb 2025 03:32:05 GMT
Usage-Pref: tdm=y,ai=n
Content-Type: text/html

<HTML page contents omitted from this example>
```

Similarly, a website might use "robots.txt" (see Section 6.1) to indicate that search indexing is the only acceptable form of data use for all resources with a path that starts with "/article/":

```
User-Agent: *
Usage-Pref: tdm=n,search=y
Allow: /article/
```

3. Preference Expression Strings

A preference expression string is a Unicode string.

A preference expression string comprises zero or more directives, where each directive is separated by a COMMA ("," or U+2C).

3.1. Preference

Each preference consists of a label, an EQUALS character ("=" or U+3D), and value of either "y" (U+79) or "n" (U+6E).

Preferences that do not conform to this syntax are ignored and have no effect.

Some whitespace -- specifically spaces (U+20) and horizontal tabs (U+9) -- are ignored to make authoring easier. Whitespace is ignored before and after both labels and values.

3.2. Labels

Each label describes how an automaton might use content. A set of core labels is defined in this document; see Section 4.

A label that is not understood by an automaton are ignored. Similarly, labels that do not apply to a usage are ignored.

3.3. Values

Each preference includes one of two values: "y" or "n".

A value of "y" indicates a preference to allow the associated usage, unless there is a label for a more specific usage that has a value of "n".

A value of "n" indicates a preference to deny the associated usage, unless there is a label for a more specific usage that has a value of "y".

Any other value causes a preference to be ignored.

3.4. Duplicated Labels

Preferences with duplicated labels are permitted. This can happen when multiple preference expressions are combined; see Section 5.4.

If multiple preferences include the same label, any preference with a value of "n" applies. Failing that, any preference with a value of "y" is used. Preferences that cannot be parsed successfully as either "y" or "n" are ignored.

For example, the following indicates a preference to deny artificial intelligence uses.

ai=y,ai=n,ai=y,unknown=y

The order of duplicate labels has no effect on the outcome, until the preference expression string exceeds the length permitted by the automaton processing it.

3.5. Specificity

A label might be defined to be more specific than another label.

The value for the most specific applicable label applies, even if the more general label has a contradictory value.

For example, the following indicates a preference to allow a use in generative AI applications, but not for other AI applications:

garbage!!!,genai=y,ai=n

The order of preferences and the presence of invalid values has no effect on the outcome.

4. Usage Labels

This section defines a limited taxonomy of usages, with just the following broad usage labels:

tdm:
: The "tdm" label relates to all forms of automated processing of content. The acronym TDM stands for "text and data mining". This is a broad category that refers to the practice of the automated extraction of any sort of information from content.

ai:
: The "ai" label describes usage for all forms of artificial intelligence or machine learning. This includes both the training of models using the content and the use of a model that has been trained using the content. The "ai" label is more specific than "tdm".

genai:

The "genai" label describes usage for artificial intelligence or machine learning models that are able to produce content. That content does not need to be similar in form to the content that a preference expression is applied. This includes both the training of models using the content and the use of a model that has been trained using the content. The "genai" label is more specific than "ai".

search:
The "search" label describes a use of content for developing a search index and the use of those indexes to find content. The "search" label is more specific than "tdm".

| Note that although search applications often use machine
| learning, "search" is not more specific than the "ai" label.

4.1. Defining New Labels

The set of labels can be expanded, but the core set is intended to be small. The most important goal is to ensure that each label is clearly understood and widely recognized. A proliferation of choices makes it harder to choose the right label to use and increases the chances that a label is not recognized by an automaton.

An entity processing a preference expression only uses preferences that are known to it. Preferences that contain labels unknown are ignored. This potentially allows for the inclusion of preferences for new usages.

Because new labels might not be recognized, it might be necessary for preference expressions to include values for more general labels in addition to the new and more specific label.

For example, if a new "example" usage is defined to more specific than the existing "tdm" label, there are four potential expressions. Depending on whether an automaton is updated to support the new label, the outcome differs, as shown in Table 1.

example	tdm	updated	old
n	n	DENIED	DENIED
n	y	DENIED	ALLOWED
y	n	ALLOWED	DENIED
y	y	ALLOWED	ALLOWED

Table 1: Backward Compatibility for New Labels

If automatons that might use content in the new way can also be expected to understand the new label, then the new label can be safely used. Otherwise, there is a risk that automatons infer permission based on more general preferences.

Section 8 describes a process for registering new labels.

4.2. Label Characters

A preference expression cannot include certain characters when carried in specific protocols (see Section 6):

* A "#" (U+23) character is interpreted as a line ending in the "robots.txt" format Section 6.1.

* The structured field dictionaries used for the auto-usage field in HTTP Section 6.2 limit character repertoire to lowercase alphabetic characters ("a" through "z"), digits ("0" through "9"), "_", "-", ".", and "*".

* Line ending characters (U+10 and U+13) and control characters (U+00 through U+31) are not special in this format. However, some systems could be incapable of authoring, conveying, or rendering them without alteration.

The format does not prohibit the use of these characters. However, any new labels intended for wide compatibility SHOULD use only lowercase alphabetical characters.

5. Processing Preference Expression Strings

This section details an example algorithm for processing of preference expressions.

An automaton that receives a preference expression uses the following algorithm, or any process with an equivalent outcome, to determine whether a particular usage is ALLOWED or DENIED according to that expression.

1. The automaton parses the preference expression and produces a record of values for each label known to it following the process in Section 5.1.

2. The automaton selects the labels that apply to its current usage, including labels at all levels of generality. These are passed to the determination process in Section 5.2 to determine whether the usage is ALLOWED or DENIED.

5.1. Parsing

The process of parsing a preference expression and updating the record of values, is as follows:

1. The automaton initializes a record one variable for each of the labels that is known to it with a value of UNKNOWN. Include labels in this record that are more general, even if the specific usage is properly described by that subset (for example, include "tdm" in addition to "search" for a simple search indexing function).

2. The expression is split into a sequence of preferences by splitting the string at each COMMA (",", U+2C).

3. Each preference is then processed in turn:

   a. The preference is split into a label and a value at the first EQUALS ("=", U+3D). Preferences that do not contain an EQUALS ("=", U+3D) are skipped and processing continues with the next preference.

   b. Spaces (U+20) and tabs (U+9) are trimmed from the start and end of each label and value. (ISSUE: move this up a step for SF compatibility and easier parsing?)

   c. If the label is not understood by the automaton or the label does not apply to the current usage, it is ignored and the processing continues with the next preference.

   d. If the value is the string "n" (a single U+6E), the record for that the corresponding label is set to NO.

e. If the value is the string "y" (a single U+79), and the current record for the corresponding label is not set, that record is set to YES.

5.2. Determination

When values have been recorded for known labels, the automaton selects the labels that applies to a specific usage.

This includes more general labels. For example, a "genai" application would include both "ai" and "tdm" labels.

The following process can be followed to determine whether that usage is ALLOWED or DENIED according to a preference expression.

1. Starting with the most general labels, iterate over the recorded value for each label.

   a. For each of the labels that are more specific than this label, replace any value of UNKNOWN for that more specific label with the value from this (more general) label.

   b. If all aspect of the usage is covered by more specific labels discard the more general label from the record.

2. If the value for any of the remaining labels is UNKNOWN, the automaton substitutes a YES or NO value based on its configured policy for missing preferences for the corresponding label.

3. If the value for any of the remaining labels is NO, resolve that the indicated preference regarding the usage is DENIED.

4. Otherwise, resolve that the indicated preference regarding the usage is ALLOWED.

In addition to understanding which labels apply to a given usage, automatons need to have a default policy for each of those labels.

5.3. Optimization and Length Limits

The parsing of preference expressions can be implemented in a streaming fashion, requiring minimal state beyond that needed to track the value of applicable labels.

A preference expression of arbitrary length can be processed efficiently. However, implementations MAY choose to stop processing expressions at any point once at least 1000 characters have been processed.

5.4. Multiple Preference Expressions

An automaton that is consuming content might encounter multiple preference expressions that relate to the same piece of content. For example, preference expressions might be carried by both "robots.txt" (Section 6.1) and content metadata (Section 6.3).

Multiple preference expressions can be combined by simple concatenation in any order. A single comma (",") is inserted between each expression.

For an identical outcome, the parsing process (Section 5.1) can be run separately for each expression, updating the same record of values rather than starting anew for each.

5.5. Multiple Usages

An automaton that might use content for multiple purposes can parse preference expressions once.

The parsing process (Section 5.1) can be run, recording the value of all labels that apply to any of the potential usages.

The state that is produced from parsing can be retained and applied (Section 5.2) to each different usage.

6. Carrying Preference Expressions

A preference expression might be propagated by multiple different mechanisms. This provides content creators and distributors choice in how they manage the signaling of their preferences.

6.1. The "robots.txt" Format

Use of "robots.txt" , or the Robots Exclusion Protocol [ROBOTS], provides automated crawlers with information about what resources can be gathered.

This is a file that is served in a well-known location by HTTP servers.

An extension is defined for this format to allow for carrying simple usage preference strings. A rule name of "usage" is added to each group, appearing between the startgroupline and rule lines as follows:

```
group = startgroupline
        *(startgroupline / emptyline)
        *(usage / emptyline)
        *(rule / emptyline)
usage = *WS "usage" *WS ":" usage-pref-exp EOL
usage-pref-exp = *UTF8-char-noctl
```

Notes:

* The label "usage" is case insentive, but the preference expression is case sensitive.

* Preference expressions will be truncated at the first "#" character (U+23); see Section 4.2. Labels that include this character can be omitted.

6.2. HTTP Header Field

An HTTP Response Header field called Content-Usage is defined.

This field is defined as a structured field dictionary, as defined in Section 3.2 of [RFC9651].

Dictionary members MUST all include a single token value (Section 3.3.4 of [RFC9651]) that is either "y" or "n".

The Content-Usage field applies to the content of a request or response, not the resource or representation.

The Content-Usage field can be used in requests to have the preference apply to the content of the request. Servers could retain a copy of preferences if the content of a request is used to answer later requests. For example, the content of a PUT request that is used to answer subsequent GET requests. Servers SHOULD reject requests that include Content-Usage unless the same or compatible preferences can be provided to entities that might obtain the included content. Obviously, servers that have not been updated to this specification will not.

Content-Usage does not have any special effect on caching.

The value of this field MAY be first parsed using a structured field parser. This implies the following restrictions:

* Structured field parsers do not permit whitespace between dictionary key, the "=" separator, and value. Therefore, this additional space MUST be removed before creating the field.

* Dictionary members are tokens, not Booleans (Section 3.3.6 of [RFC9651]). Boolean members MUST be ignored.

* Dictionary members with parameters MUST be ignored.

6.3. Content Metadata

Specific formats can define how preference expressions are carried in content metadata.

This document does not define any such format.

| Question: Should we define how to use HTML meta, or rely on
| http-equiv?

7. Security Considerations

This document defines a preference mechanism, not a security feature.

The primary consideration for security is the correct and efficient implementation of parsing. Errors in parsers have been known to be exploited by actors that produce malicious input to effect one of two common types of problem:

* buffer overruns, where malicious input is formed to enable remote code execution

* denial of service, where malicious input is constructed to be difficult to process in order to exhaust resources

The algorithms in Section 5 are not immune to these issues if incautiously implemented by an automaton. They are exemplary only and are biased toward being comprehensible rather than being secure.

Preference strings could include characters that can create confusing or misleading renderings. For instance, invisible characters or U+8 (backspace) could be used to create a misleading representation. These characters can be avoided when authoring preference expressions and replaced when displaying them (using U+FFFD for example).

8. IANA Considerations

This document defines a new registry for labels (Section 8.1) and an HTTP header field (Section 8.2).

8.1. Preference Label Registry

This document establishes a registry that lists Labels for Preference Expressions for Automated Use of Content with IANA. The policy for new registrations is Specification Required [RFC8126].

Each entry has the following fields:

Label:
: a Unicode string (see Section 4.1 for advice on selecting values)

Definition:
: a short description of the automated usage that the label applies to

Narrows:
: a list of other labels that this label is more specific than

Specification:
: a link to a publicly accessible specification that contains a more complete definition

Status:
: either is "provisional" or "permanent" (see below)

Date:
: the date of the last update to the registration

Change Controller:
: the entity responsible for the specification

Contact:
: contact details for the registrant

8.1.1. Registration Guidance

An ideal label for a new registration is short, descriptive, and memorable. Lowercase characters (U+61 through U+7A) are most compatible; see Section 4.2.

Provisional registrations can be used by entities other than a change controller to register labels in order to avoid registration collisions. Provisional registrations can omit fields. However, provisional labels can be removed if details are not provided within 12 months. Removals are subject to approval by the IESG.

Designated experts are advised to discourage new permanent registrations in favor of provisional registrations. Once utility and -- critically -- understanding of labels has been demonstrated, provisional registrations can be upgraded to permanent status.

Designated experts are expected to help ensure that registrations correctly identify which labels are more general. This could include updates to existing labels, to reference a new registration as being more general.

Designated experts are expected to deny registration requests for large numbers of labels, labels that are unclear in purpose, labels that are too similar to existing labels, labels that are long, and labels that promote proprietary products or unproven concepts.

Expert decisions can be appealed to the IESG.

8.1.2. Initial Registry Contents

Table 2 tabulates the other fields for registrations of labels from Section 3.2.

Label	Definition	Narrows
tdm	Any automated process that extracts information from content	
ai	Training or use of any machine learning system (or AI)	tdm
genai	Training or use of machine learning (or AI) that can generate content	ai
search	Generation of search index or use for search applications	tdm

Table 2: Initial Label Registrations

Initial registrations in this registry also include the following values:

Specification: this document
Status: permanent
Date: the date of publication of this document
Change Controller: IETF
Contact: IETF AI-PREF WG (ai-control@ietf.org)

8.2. HTTP Content-Usage Field Registration

An HTTP field named Content-Usage is registered in the Hypertext Transfer Protocol (HTTP) Field Name Registry, following the procedures in Section 18.4 of [HTTP]. The following values are registered:

Field Name: Content-Usage
Status: permanent

Reference: this document
Comments: (none)

9. References

9.1. Normative References

[HTTP] Fielding, R., Ed., Nottingham, M., Ed., and J. Reschke, Ed., "HTTP Semantics", STD 97, RFC 9110, DOI 10.17487/RFC9110, June 2022, <https://www.rfc-editor.org/rfc/rfc9110>.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/rfc/rfc2119>.

[RFC8126] Cotton, M., Leiba, B., and T. Narten, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 8126, DOI 10.17487/RFC8126, June 2017, <https://www.rfc-editor.org/rfc/rfc8126>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/rfc/rfc8174>.

[RFC9651] Nottingham, M. and P. Kamp, "Structured Field Values for HTTP", RFC 9651, DOI 10.17487/RFC9651, September 2024, <https://www.rfc-editor.org/rfc/rfc9651>.

[ROBOTS] Koster, M., Illyes, G., Zeller, H., and L. Sassman, "Robots Exclusion Protocol", RFC 9309, DOI 10.17487/RFC9309, September 2022, <https://www.rfc-editor.org/rfc/rfc9309>.

9.2. Informative References

[REP-PURPOSE] Illyes, G., "Robots Exclusion Protocol User Agent Purpose Extension", Work in Progress, Internet-Draft, draft-illyes-rep-purpose-00, 18 October 2024, <https://datatracker.ietf.org/doc/html/draft-illyes-rep-purpose-00>.

[VOCAB] Vaughan, T., "Vocabulary for Expressing Content Preferences for AI Training", Work in Progress, Internet-Draft, draft-vaughan-aipref-vocab-00, 20 January 2025, <https://datatracker.ietf.org/doc/html/draft-vaughan-aipref-vocab-00>.

Acknowledgments

This document is informed by discussions at the AI-CONTROL workshop.

Drafts from Gary Ilyes ([REP-PURPOSE] and Thom Vaughan [VOCAB] helped inform some of the choices. This document aims to be more complete than the former and simpler than the latter.

Author’s Address

Martin Thomson
Mozilla
Email: mt@lowentropy.net