

Strawman proposals for extension of the Robots Exclusion Protocol Part 2: Extension of Robots META Tags and other techniques for embedding permissions in HTML content

A component of the ACAP Technical Framework

Francis Cave, ACAP Technical Project Manager

Draft for pilot testing, Version 2, 15 October 2007

Introduction

ACAP (Automated Content Access Protocol) is being developed as an open industry standard to enable the providers of all types of content to communicate permissions information (relating to access to and use of that content) in a form that can be readily recognized and interpreted by a search engine (or any other intermediary or aggregation service), so that the operator of the service is enabled systematically to comply with the individual publisher's policies. ACAP will provide a technical framework that will allow publishers worldwide to express access and use policies in a language that machines can read and understand

While permissions are often assigned to whole classes of content, there is nevertheless a need to associate permissions with a specific content item, overriding or adding to permissions assigned to a class to which the specific item belongs. Permissions associated with whole classes of content may be most efficiently expressed in resources separate from the content itself – for example, in a robots.txt file. However, permissions associated with a specific resource can in some cases only be effectively expressed when embedded in the content item itself.

Various techniques are already in use for communicating permissions embedded within HTML content. The most widely-used technique is known as Robots META Tags, which with the robots.txt format is part of the Robots Exclusion Protocol. The interpretation of Robots META Tags is supported by some at least of the major search engines.

This document is the second part of a two-part proposal that addresses the issue of how the requirement of ACAP to communicate permissions consistently and unambiguously can be reconciled with the need to continue to use REP, by setting out strawman proposals for the extension of REP.

Other techniques include the use of the HTML `class` attributes to embed permissions in arbitrary elements within content, although there isn't such widespread recognition of these as there is of Robots META Tags.

NOTE. This discussion paper uses the syntax conventions of HTML 4.0, but ACAP will provide equivalent support for embedding permissions in XHTML documents as in HTML documents.

1. Robots META Tags

As reported on the Robots Exclusion Protocol website¹, Robots META Tags originated at a conference in May 1996 organised by the World Wide Web Consortium². The outputs from that conference include a BOF ('Birds of a Feather') report on "spidering". This report recommended two uses of the HTML META tag to

¹ See <http://www.robotstxt.org/wc/exclusion.html>.

provide information to crawlers: one for conveying permissions information and one for conveying descriptive information that could be used as an aid to indexing.

The HTML META tag is a header-level tag – i.e. it is only allowed inside the HTML HEAD element and not inside BODY. It is designed to carry metadata about the entire HTML page.

The META tag represents an empty HTML element that can take up to four “attributes”, of which the two that are used in Robots META Tags are `name` and `content`. The other two are 'scheme', which is used to convey the name of a scheme (controlled vocabulary) to be used when interpreting the value of the 'contents' attribute and is not used in Robots META Tags; and 'http-equiv', which is used to provide information to the web server for use in constructing fields to include in an HTTP response header for this HTML page, also not used in Robots META Tags.

In Robots META Tags the `name` attribute always has the value "robots" (not case sensitive). The `content` attribute contains any of the values "index", "noindex", "follow", "nofollow", "all" or "none". Values can be combined, separated by commas, e.g. "index, nofollow".

The interpretations of the values of the `content` attribute are documented as follows:

- "index" / "noindex" mean: "search engines are / are not welcome to index this page"
- "follow" / "nofollow" mean: "search engines are / are not welcome to follow links from this page to find other pages"
- "all" means the same as the combination of "index, follow"
- "none" means "noindex, nofollow"

The absence of a value is permissive, i.e. "noindex" on its own is interpreted as "noindex, follow", "nofollow" on its own is interpreted as "index, nofollow", and the absence of a Robots META Tag on a page is interpreted as "all" or "index, follow".

1.1. Can ACAP use Robots META Tags as they are?

Robots META Tags have precise and unambiguous definitions and could be usable without modification for their original intended use.

However, Robots META Tags are strictly limited in the permissions that they can express, to permit or prohibit the usages "index" and "follow (link)". Also, the format

² See <http://www.w3.org/Search/9605-Indexing-Workshop/>.

as defined is incapable of expressing qualifications of these usages, and also is incapable of defining the scope of the permission to be anything less than the entire HTML page.

1.2. What can be done to change Robots META Tags?

As with robots.txt, any changes to Robots META Tags will have to be agreed to by the major search engines. The approach that should be adopted by ACAP is to put forward proposals for extensions to the existing format.

1.3. Syntax of ACAP extensions to the Robots META Tags format

NOTE – Formal syntax definitions may be found at the end of this section.

The Robots META Tags format as defined on the Robots Exclusion Protocol website recognises only one value of the `name` attribute of a META tag, which is `"robots"`. It is understood that some search engines recognise the name of a specific crawler in place of `"robots"`. ACAP proposes to make use of this format extension.

In order to extend Robots META Tags without interfering in the interpretation of existing META Tags, it is proposed that all ACAP permissions, prohibitions and definitions be distinguished by an initial token `ACAP` at the start of the value of the content attribute, e.g.

```
<META name="robots" content="ACAP ...">
```

or, for a named crawler:

```
<META name="crawler-name" content="ACAP ...">
```

The order in which ACAP permissions and prohibitions are expressed in META Tags is not significant.

1.3.1. ACAP permission

The proposed syntax for expression of an ACAP permission as the value of the content attribute is as follows:

```
ACAP [usage-purpose-pattern] allow-usage  
[fragment-specification]
```

where the syntax of *usage-purpose-pattern* and *usage* are as proposed in Part 1 of these proposals.

The square brackets are not part of the proposed syntax, but are used here (and likewise in subsequent syntax examples) to indicate that the inclusion of *usage-purpose-pattern* and *fragment-specification* are optional.

The inclusion of *fragment-specification* indicates specific fragments within the page to which the permission applies and is a specification of identified fragments with the following syntax:

```
id-list= fragment-identifier ...
```

where *fragment-identifier* is repeatable and must follow the syntax rules defined for HTML identifiers.

1.3.2. ACAP prohibition

The proposed syntax for expression of an ACAP prohibition as the value of the content attribute is as follows:

```
ACAP [usage-purpose-pattern] disallow-usage
```

An ACAP prohibition applies to the whole page unless a preceding ACAP permission specifies exceptions for specified fragments within the page.

1.3.3. Resolving conflicts between overlapping permissions and prohibitions applied to resources and fragments

If two META Tags contains conflicting permission and prohibition, both addressed to the same named crawler or both addressed to all crawlers, and both applying to the same usage of the resource as a whole, the usage shall be interpreted as prohibited.

If a crawler is capable of interpreting permissions and prohibitions applied to fragments, a permission or prohibition of a specified usage applied to a specified fragment within a resource shall *for the specified fragment only* override any permission or prohibition of the *same usage* applied to the resource as a whole.

If a crawler is capable of interpreting permissions and prohibitions applied to fragments and if a permission or prohibition of a specified usage is applied to a specified fragment within a resource, the permission or prohibition applies to the whole fragment and any permission or prohibition of the *same usage* applied to a fragment nested within the specified fragment shall be ignored.

A crawler may not be capable of interpreting permissions and prohibitions applied to fragments, in which case the crawler shall ignore all such permissions and prohibitions and only interpret the permissions and prohibitions that apply to the resource as a whole. Content owners should bear this in mind when specifying permissions and prohibitions on fragments and specify accordingly the permissions and prohibitions on the resource as a whole.

1.3.4. ACAP definitions

ACAP definitions, as proposed in Part 1 of these proposals, may be specified within an HTML page using META Tags. The following syntax is proposed:

```
<META name="ACAP-definition-name"
value="ACAP-definition-content">
```

where *ACAP-definition-name* has one of the following values:

```
ACAP-qualified-usage
ACAP-composite-usage
```

and *ACAP-definition-content* has a value in accordance with the syntax proposed for ACAP definitions in Part 1 of these proposals.

All the usages and associated usage qualifiers that are specified in Part 1 of these proposals may be used in definitions of qualified usages in META Tags, although many are only meaningful if applied to the whole resource, and not to identified fragments within it. The following table shows additional usage qualifiers that are not suitable for use in robots.txt, but may be used in META Tags.

Usage name	Qualifier type	Usage qualifier value	Description
index	location	<i>URI</i>	if permission applies to the resource when it is located at the specified URI
preserve	location	<i>URI</i>	if permission applies to the resource when it is located at the specified URI
present	location	<i>URI</i>	if permission applies to the resource when it is located at the specified URI

NOTE – The location qualifier may be repeated if the permission applies to the resource when it is at any of a number of locations.

1.3.5. Formal syntax definition

A formal definition of the syntax of these proposed extensions is given here, using the ABNF notation defined in IETF RFC 2234. “URI” and “relative-part” are defined in IETF RFC 3986. The token `<ACAP-meta-tag-name>` defines a syntax extension for the value of the `name` attribute on a META element in the header of an HTML (or

XHTML) resource. The token `<ACAP-meta-tag-content>` defines a syntax extension for the value of the `content` attribute on a META element.

The syntax tokens `<crawler-name>`, `<ACAP-usage-name>`, `<qualified-usage-name>`, `<composite-usage-name>`, `<local-usage-name>`, `<usage-purpose-pattern>` and `<name>` are defined in Part 1 of these proposals. The token `<WSP>` is defined in IETF RFC 2234.

`ACAP-meta-tag-name` = "robots" / crawler-name / ACAP-definition-name

`ACAP-definition-name` = "ACAP-qualified-usage" / "ACAP-composite-usage"

`ACAP-meta-tag-content` = "ACAP" 1*WSP (ACAP-permission / ACAP-prohibition / ACAP-qualified-usage-definition / ACAP-composite-usage-definition)

NOTE – An `<ACAP-qualified-usage-definition>` may only occur in the value of the `content` attribute if the `name` attribute contains "ACAP-qualified-usage". Similarly, an `<ACAP-composite-usage-definition>` may only occur in the value of the `content` attribute if the `name` attribute contains "ACAP-composite-usage"

`ACAP-permission` = [usage-purpose-pattern 1*WSP] "allow-" (ACAP-usage-name / local-usage-name) [fragment-specification] 1*(1*WSP qualifier-specification)

`fragment-specification` = "id-list=" *WSP fragment-id *(1*WSP fragment-id)

`fragment-id` = name-start-char *name-char

`ACAP-prohibition` = [usage-purpose-pattern 1*WSP] "disallow-" ACAP-usage-name

NOTE – A `<local-usage-name>` in an `<ACAP-prohibition>` may not refer to composite usage.

`ACAP-qualified-usage-definition` = qualified-usage-name 1*WSP ACAP-usage-name 1*(1*WSP qualifier-specification)

`ACAP-composite-usage-definition` = composite-usage-name 1*(1*WSP (ACAP-usage-name / local-usage-name))

2. Other techniques for embedding permissions in HTML content

2.1. Use of the `class` attribute

The HTML `class` attribute is available for use on all BODY elements, and is generally used to carry a CSS³ style class name. However, the `class` attribute was not designed exclusively for use in carrying style information, and can quite legitimately be used to carry permissions or other processing information.

The value of a `class` attribute must be either a single class name or multiple class names separated by spaces.

The ability to assign a `class` attribute to any BODY element is both a strength and a weakness. Its strength is that it enables permissions to be associated with any fragment of a document by simply embedding. Its possible weakness – particularly from an aggregator's perspective – is that the permissions are scattered throughout an HTML page, rather than in a single location, such as in a series of META tags.

However, when it is necessary to associate permissions with specific components within a page – for example, to indicate which sections or paragraphs in a page may be indexed and which not – it may in fact be easier to process such permissions if they are placed directly on the elements in question than if they are at the top of the document.

Let us consider the three ways that a permission could be associated with a component within a page.

If the permission is contained in an external resource, e.g. **robots.txt**, the permission has to locate the component to which it relates using one of the techniques for locating fragments in an HTML (or XHTML) page. The aggregator's system that is processing the resource and needing to interpret the permissions that relate to it will have to check each component of the resource to determine whether any external permissions apply to it. This represents a considerable processing overhead.

If, instead, permissions are stored at a single point within the resource, e.g. in **Robots META tags**, the components must still all be checked to see if any permissions apply to them specifically. Only one resource is involved, but there is still an overhead in checking components with which no permissions are associated.

The third option is to embed the permission in the component, e.g. in a **class attribute**. In this case the permissions are only encountered when the resource as a whole is processed, but components that don't contain embedded permissions don't need to be checked.

³ Cascading Style Sheets – a standard format for specifying the style of presentation of an HTML document.

It would therefore seem that, while the use of `class` attributes would not be appropriate for carrying all kinds of permissions, their use for carrying component-level permissions might be the most appropriate way of associating permissions with specific components, particularly where these permissions could be used in relation to usages that involve derivations from the entire content of the resources (e.g. indexing, snippet generation)⁴.

The syntax for the `class` attribute is quite limited, which suggests that its use will only be suited to simple usage permissions and prohibitions. The proposed approach is for ACAP permissions to adopt a naming convention involving the pseudo-namespace prefix `acap:`, e.g.

```
<div class="abstract acap:allow-usage">...
```

where "abstract" is publisher-defined but "usage" is the name of an ACAP-defined usage or a locally-defined (qualified or composite) usage (see 1.3.4).

2.2. Syntax of proposed ACAP extensions to the use of the `class` attribute

It is proposed that only usage permissions and prohibitions be embeddable in `class` attributes. Permissions could make reference to qualified or composite usages defined in META Tags in the same resource, as well as appropriate standard ACAP usage verbs. Usage purpose patterns may not be included.

```
class-attribute-permission = "acap:allow-"  
                             (ACAP-usage-name / local-usage-name)  
  
class-attribute-prohibition = "acap:disallow-" ACAP-usage-name
```

⁴ Technical objections to this approach would undoubtedly include the fact that not all HTML pages are "well-formed", i.e. they don't always use tags in accordance with the HTML (or XHTML) specifications. It may be that the expression of permissions at component level would have to be conditional on the use of XHTML, rather than legacy HTML formats, to ensure that the content is well-formed.