Communicating access and usage policies to crawlers using extensions to the Robots Exclusion Protocol
Part 1: Extension of robots.txt format

A component of the ACAP Technical Framework

Implementation Version 1.0 (corrected), 30 November 2007

Document history

Version	Release date
Pilot Version 1	2007-09-15
Pilot Version 2	2007-10-15
Pilot Version 3	2007-11-19
Implementation Version 1.0	2007-11-26
Implementation Version 1.0 with corrections	2007-11-30

Table of contents

Changes relative to previous version

1. General changes

1.1 URI scheme name “acap” changed to “the-acap” to avoid clash with existing registered scheme of the same name.

1.2 Extra trailing slash added to resource path patterns that represent directories (folders), to avoid the ambiguity that they might be interpreted as resources (files).

1.3 Resource type ‘extract’ removed, due to lack of a clear definition or use.

1.4 Copyright line added on cover page.

2. Specific changes

2.1 In Section 2.3.1 the final note has been extended to recommend that a crawler name be the longest string that a crawler will recognise, to avoid the same name being recognised by two crawlers.

2.2 In Section 2.5.5.2.5 the resource type link is marked as not ready for implementation.

2.3 In Section 2.8 a final paragraph has been added to make it clear that pattern matching is case-insensitive.

2.4 In Section 5 the hypertext links to other ACAP Technical Framework documents on the ACAP website have been corrected.

1 Introduction

ACAP (Automated Content Access Protocol) is being developed as an open industry standard to enable the providers of all types of content (including, but not limited to, publishers) 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.

The Robots Exclusion Protocol (REP) is the formal name for what is currently the most widely-used method of communicating permissions to web crawlers (also frequently referred to as ‘robots’ or ‘spiders’) [1]. This method of communication is in two parts: a format for a file called 'robots.txt' that contains machine-readable statements of which sets of resources on a website may or may be crawled; and a format for embedding permissions in HTML page headers, called Robots META Tags.

This document is Part 1 of a two-part specification of a method of communication based upon proposed extensions to the Robots Exclusions Protocol. This part describes proposed extensions to the robots.txt format to meet the requirements of a series of use cases tested in the ACAP pilot project. The format proposed by this document has been tested against several important use cases and is considered to be ready for implementation for most use cases that involve communication to search engine crawlers of access and usage policies relating to publicly-accessible online content. The format has also been tested for use cases that involve similar communication of access and usage policies relating to online content that is *not* publicly-accessible, but it is recognised that further clarification and extension of the format may be needed in this area.

A companion document forms Part 2 of the specification[2], which contains proposed extensions to the Robots META Tags format and other techniques for using embedded usage permissions and prohibitions in HTML documents.

Both this document and its companion use access and usage terminology that is defined in the ACAP Dictionary of Access and Usage Terminology[3].

1.1 Implementation status of proposed ACAP extensions

The proposed extensions to REP in both parts of this two-part specification are labelled according to their current implementation status, using the following colour coding scheme.

Most features are ready for implementation now for general use in crawler communication and are therefore not labelled.

● Features that are ready for implementation now, but only for use in crawler communication by prior arrangement, are labelled with an amber spot. These represent a minority of extensions for which there are possible security vulnerability or other issues in their implementation on the web crawler side, such as creating possible new opportunities for cloaking or Denial of Service attack. It is anticipated that these extensions will only be implemented in established and trusted business relationships between web server and web crawler.

● Features that require further development and testing prior to being ready for implementation, but included here because they are considered to be stable forms of expression, are labelled with a red spot.

DISCLAIMER – The proposals for extension of the Robots Exclusion Protocol (REP) contained within this two-part specification are merely proposals and not yet adopted by most crawler operators. All features, especially those that are marked as not yet ready for implementation, may change or be withdrawn without notice. All implementations of these proposals are entirely at the implementer's own risk, and no particular outcome is guaranteed. Implementation of these proposals in either robots.txt or META tags may cause some crawlers to cease to crawl some resources, while others may misinterpret or ignore the proposed extensions to REP.

2 Description of ACAP extensions to the robots.txt format

2.1 Overview

This document proposes extensions to the robots.txt format to express a content owner's policy for allowing or denying crawlers access to and use of their online content. These extensions do not replace the existing robots.txt format, but enable unambiguous expression of permissions, both unqualified and qualified by a range of restrictions[1], and outright prohibitions as to what a crawler and associated automated follow-on processes may or may not do with specified content.

Some of these extensions may appear, at first sight, to duplicate the functionality of the existing robots.txt format. The reason for this is that the precise semantics of the existing format are not defined, so, in order that a content owner's policy can be expressed without ambiguity, it is necessary to adopt new forms of expression. It is

[1] Future revisions of this document are expected to include a method for positively expressing that there is no restriction on a permission. This document only defines qualifiers that are used to communicate restrictions on permissions.

anticipated that at some stage such apparent duplications will be eliminated by formal standardization of the Robots Exclusion Protocol.

The ACAP extensions are designed to be used within robots.txt files that also contain permissions and prohibitions using the existing format. A single robots.txt is typically used to communicate with a large number of crawlers (hundreds of crawlers are known to read robots.txt files), and it will take time for crawlers to be programmed to recognise and use the proposed ACAP extensions.

A typical robots.txt file using these extensions will be in two parts, as illustrated by the example below. The first part will largely contain existing permissions and prohibitions, apart from a special comment at the very start, indicating that ACAP proposed extensions are being used. The second part will contain ACAP permissions and prohibitions organised into a series of crawler records that may be preceded by a series of definition fields.

```
##ACAP version=1.0
# Legacy robots.txt content…

User-agent: *
Disallow: /

User-agent: named-crawler
Allow: /index.html
Allow: /public/
Allow: /promotion/
Allow: /news/

# Un-comment the line below, if crawlers capable of understanding
# ACAP records are to ignore conventional records
# ACAP-ignore-conventional-records

# ACAP local definitions
# Resources found in three directories are crawlable
ACAP-resource-set: crawlable /public/ /promotion/ /news/

# On this site ‘cache’ means ‘preserve (store) until re-crawled’
ACAP-qualified-usage: cache preserve time-limit=until-recrawled

# The same usages are permitted for all resources in the specified
# resource set, so we can define a composite usage
ACAP-composite-usage: basic-usages crawl index present

# Crawlers in general are prohibited to crawl this site
ACAP-crawler: *
ACAP-disallow-crawl: /

# Named crawler may crawl, index and display the content of /public/ ...
```

```
ACAP-crawler: named-crawler
# All my usages are permitted for the specified resource set...
ACAP-allow-(basic-usages): the-acap:resource-set:crawlable
# which is equivalent to permitting the three separate usages, which are
# commented out here...
# ACAP-allow-crawl: the-acap:resource-set:crawlable
# ACAP-allow-index: the-acap:resource-set:crawlable
# ACAP-allow-present: the-acap:resource-set:crawlable

# ...but may only preserve copies in the locally-defined sense
# ACAP-allow-(cache): the-acap:resource-set:crawlable
```

NOTE – Throughout this two-part specification examples are presented in monospaced text on a pale green background.

2.2 General

ACAP access and use policies in 'robots.txt' are expressible in one or more ACAP records. An ACAP record defines a set of permissions, prohibitions and action requests addressed to one or more crawlers. Each ACAP record comprises a number of fields, each of which is represented by a single line of the robots.txt file. The fields in a record may be separated by blank lines and comments[2].

An ACAP record may, if appropriate, contain sub-records that group permission and prohibition fields according the usage purpose to which they relate. An ACAP record can contain a mixture of sub-records associated with specified purposes and general permissions and prohibitions that are not specific to a particular usage purpose, in which case the general permissions and prohibitions (and any action requests) occur before the sub-records in the ACAP record.

The order in which ACAP fields occur within an ACAP record is only significant in determining which fields belong to each record and, within a record, which fields belong to sub-records, if any. All ACAP records and all conventional records must be interpreted by a crawler to determine which to apply to any given resource, unless the robots.txt file contains a directive that conventional records are to be ignored, in which case only ACAP records are interpreted.

In cases where fields within a record have contradictory or overlapping interpretations, a mechanism for resolving such conflicts is proposed below – see Section 2.4.5.

[2] A comment in a robots.txt file is any line or part of a line that begins with a hash symbol #. The hash symbol indicates to a crawler that it should not process any further characters in that line, and is therefore reserved for that function.

The interpretation of a field may depend upon interpretation of the definition of a qualified usage, composite usage or resource set. Qualified usage and composite usage definitions and resource set definitions must precede all ACAP records.

An ACAP record may either be addressed to one or more named crawlers or to "any crawler". Permissions and prohibitions contained in a record addressed to one or more named crawlers override permissions with the same usage purposes (if any) and usage types and matching the same resource(s) contained in a record addressed to "any crawler".

2.3 ACAP records and sub-records

An ACAP record must contain fields in the following order (comments and blank lines being ignored):

- one or more crawler identification fields, identifying the crawlers to which the permissions associated with this record are addressed

followed by *either*

- an ACAP permissions reference field, directing the identified crawler(s) to another resource containing the permissions relevant to those crawlers – in which case no further fields (other than comments and blank lines) may occur in the record

or a sequence of

- one or more permission, prohibition or action request fields applicable to that crawler not associated with any specific usage purpose
- zero or more sub-records containing permissions / prohibitions associated with specific usage purposes.

or a sequence of one or more sub-records containing permissions / prohibitions associated with specified usage purposes.

An ACAP record terminates when either a crawler identification field is encountered following some other field allowed in records (ignoring comments and blanks) or the end of the file is reached, whichever occurs first.

A sub-record contains one or more usage purpose pattern fields followed by one or more permission or prohibition fields. A sub-record may not contain action request fields.

A sub-record terminates at the first occurrence following a permission or prohibition field of one of the following:

- a usage purpose pattern field (indicating the start of the next sub-record)
- a crawler identification field (indicating the start of the next record)

- the end of the file.

Records, sub-records and fields within them may be interspersed with comments and blank lines, which can be ignored for the purposes of automated interpretation.

2.3.1 Crawler identification field

An ACAP record begins with one or more crawler identification fields. Each crawler identification field specifies a single crawler to which the content of the ACAP record is addressed. By including multiple crawler identification fields, a single ACAP record may be addressed to multiple crawlers.

NOTE – A crawler identification field is semantically equivalent to a user agent field in an existing robots.txt file. The purpose of making the distinction syntactically is solely to make it clear whether a record is an ACAP record or a conventional record, because both may be included in a single robots.txt file.

A crawler identification field has the following syntax:

```
ACAP-crawler: crawler-name
```

where *crawler-name* is either a string that identifies a specific crawler or an asterisk `*`, which is interpreted to mean "any crawler".

For example, an ACAP record that addresses a crawler named `searchbot` would contain the following crawler identification field:

```
ACAP-crawler: searchbot
```

An ACAP record that addresses "any crawler" begins with crawler identification field:

```
ACAP-crawler: *
```

and must not contain any other crawler identification fields.

NOTE – The string used to identify a crawler by name is normally some or all of the string used by that crawler in a User-Agent header field in an HTTP request, as specified in IETF RFC 2616[4], and is not case-sensitive. Some crawlers may treat crawler names as patterns and match any that happen to match a sub-string of their usual User-Agent header field string, so it it recommended that a crawler name be the longest possible string that a crawler will recognise as their own name.

2.3.2 ACAP permission reference definition field

An ACAP permission reference definition links to a resource containing ACAP permissions in robots.txt format (generally using ACAP extensions) that are to be read and applied in conjunction with the permissions in the file in which this reference definition field occurs. The following syntax is proposed:

```
      ACAP-permissions-reference: resource-locator
```

where *`resource-locator`* is either a path relative to the document root on the same server as the robots.txt file containing this field, or a URI for a resource located on another server.

For example, to refer a crawler named `searchbot` to a set of additional policies in `/searchbot.txt` that are specific to that crawler, the following ACAP record would be appropriate:

```
      ACAP-crawler: searchbot
      ACAP-permissions-reference: /searchbot.txt
```

To avoid circular references, a permission reference definition field may not refer to another robots.txt file that itself contains a permission reference definition field.

2.3.3 Usage purposes

A usage purpose is a specific service or process served by one or more crawlers to which the record as a whole is addressed. A sub-record begins with one or more usage purpose pattern fields, and all permission and prohibition fields that follow these usage purpose pattern fields relate to the specified usage purposes.

A usage purpose pattern field has the following syntax:.

```
      ACAP-usage-purpose: usage-purpose-pattern
```

where *`usage-purpose-pattern`* is a pattern that can be matched by a string recognised by a crawler as identifying a usage purpose. Such a string would typically be a name or URI.

For example, a usage purpose pattern field specifying a single service with name `news` would be:

```
      ACAP-usage-purpose: news
```

An example of a usage purpose pattern field specifying all services with URIs matching a pattern would be:

```
      ACAP-usage-purpose: http://news.search.*/
```

NOTE – The interpretation of a usage purpose pattern is dependent upon the crawler being programmed to recognise specific name or URI patterns. Crawler operators are expected to indicate what usage purposes they recognise, if any.

2.4 Permissions and prohibitions

Permissions and prohibitions to access and use specified resources are expressing using permission fields and prohibition fields respectively.

2.4.1 Permission field

The simplest type of permission field is one that allows a specified standard usage of a specified set of resources, without any restriction or other qualification. More complex permission fields may also express qualified or composite usages, either by containing explicit qualifiers or by reference to a separately-defined qualified or composite usage.

2.4.1.1 Unqualified permission field

An unqualified permission field is expressed using the token `allow` combined with a specific usage type and followed by a specification of the set of one or more resources to which the permission applies, viz:

```
        ACAP-allow-usage: resource-specification
```

The usage type *`usage`* is an ACAP standard usage type as specified in Section 2.5.

The string *`resource-specification`* is a resource specification as specified in Section 2.4.3.

For example, an unqualified permission field allowing the resources that match the resource path pattern `/news/` to be indexed would be expressed:

```
        ACAP-allow-index: /news/
```

2.4.1.2 Qualified permission field

A qualified permission field may be expressed in the same way as an unqualified permission field, but with one or more qualifiers following the resource specification, viz:

```
        ACAP-allow-usage: resource-specification qualifiers
```

Each qualifier in *`qualifiers`* is constructed as follows:

```
        qualifier-type-name=qualifier-value
```

where *`qualifier-type-name`* must be a valid qualifier type name for the given usage types, and *`qualifier-value`* must be a valid value for the given qualifier type. For details of the qualifier types that are valid with each standard usage type see 2.5 (at present only usages based upon `index`, `preserve` and `present` may be qualified).

For example, a qualified permission field allowing resources that match the resource path pattern `/news/` to be preserved only until they are re-crawled would be expressed:

```
        ACAP-allow-preserve: /news/ time-limit=until-recrawl
```

A permission field may contain multiple qualifiers, if appropriate, in which case these are separated by spaces.

2.4.1.3 Permission fields based upon locally-defined usages

A permission field may express a permission for a qualified or composite usage that is defined "locally", i.e. in a qualified usage definition field or composite usage definition field specified elsewhere within the same robots.txt file, or within a robots.txt file that contains a permission reference to this robots.txt file (see 2.3.2).

ACAP-allow-(*local-usage*): *resource-specification*

A locally-defined usage type name *local-usage* must correspond to the name of a qualified or composite usage that is defined in a qualified usage definition field or composite usage definition field (see 2.7). The parentheses are required around the usage type name, to avoid a possible clash of names between locally-defined usage types and ACAP standard usage types.

No explicit qualifiers may be included in a permission field for a locally-defined usage.

For example, a permission field that allows a locally-defined usage `local-usage` to be applied to resources matching the pattern `/special/*.htm` would be expressed:

```
        ACAP-allow-(local-usage): /special/*.htm
```

2.4.2 Prohibition field

A prohibition field defines a prohibited usage of a specified set of resources, and is expressed thus:

ACAP-disallow-*usage*: *resource-specification*

where *usage* is a standard usage type.

An ACAP prohibition may not refer to locally-defined qualified or composite usages, and may not contain any qualifiers.

An example of a prohibition field:

```
        ACAP-disallow-crawl: /private/
```

NOTE – In order to express a qualified (restricted) prohibition, it is always necessary to transform this into a qualified permission. In other words, instead of expressing "you are prohibited to do x except if…", it is necessary to turn this around to express "you are permitted to do x with the following restrictions…".

2.4.3 Resource specification

A resource specification is used in permissions and prohibitions to specify to which resources they apply. A resource specification is one of the following:

- a resource path pattern that matches one or more paths to resources relative to the server document root – see 2.8
- the name of a resource set – see 2.4.3.

2.4.4 Resource set

A resource set is a named set of resources defined in a resource set definition field – see 2.7.3. A resource set is referred to by a resource specification that has the following form:

`the-acap:resource-set:`*`resource-set-name`*

where *`resource-set-name`* is the name of the defined resource set.

For example, an unqualified permission to index a set of resources named indexable would be expressed:

```
        ACAP-allow-index: the-acap:resource-set:indexable
```

where an applicable resource set definition field might be, for example:

```
        ACAP-resource-set: indexable /public/ /news/ /promote/
```

2.4.5 Conflict resolution

If at least one permission field and at least one prohibition field apply to the same usage type and are applicable to the same resource, the permission or prohibition with the narrowest effective scope is applied and the others are ignored.

Determining which permission or prohibition field has the narrowest effective scope depends upon a comparison of the resource path patterns of the conflicting fields.

Qualifiers are ignored when resolving conflicts.

For two resource path patterns to match the same resource, they must have similar patterns. The pattern with the narrowest scope can be determined by the following method:

- Compare the resource path patterns character-by-character, starting with the left-most character of each pattern.
- If the characters in the first (left-most) position in each pattern are the same, move on to compare the characters in the second and subsequent positions in each pattern until either one of the patterns runs out of characters or one of the patterns contains a wildcard character (asterisk * or dollar sign $) while the other pattern contains a different character.
- If one pattern has run out of characters, the other pattern has the narrower scope.
- If one pattern contains a dollar sign $ where the other pattern contains any other character (including the asterisk *), the other pattern has the narrower scope.
- If one pattern contains an asterisk * where the other pattern contains any other character except the dollar sign $, the other pattern has the narrower scope.

In the event that a crawler is unable to determine which ACAP permission or prohibition has the narrowest scope – regardless of whether or not such determination is theoretically possible – the usage is prohibited on that resource.

2.5 Usage types and usage qualification

Five basic usage types have so far been defined:

```
crawl
follow
index
preserve
present
```

In addition to the five basic usage types, one general usage type

```
other
```

has also been defined. This is for use in communicating blanket prohibitions of all usages other than those specifically permitted.

In addition to these six usage types, a number of other usage types have been defined, based upon the basic usage type `present`, to indicate more precisely what type or resource may be presented, having been derived in some way from the resource that has been crawled. These additional usage types are:

```
present-original
present-currentcopy
present-oldcopy
present-snippet
```

```
        present-thumbnail
        present-oldsnippet
        present-oldthumbnail
        present-link
```

In each case the basic usage type `present` is followed by a used resource type.

The used resource types `snippet` and `thumbnail` are both derived from the current (i.e. most recently crawled) version of a resource. The used resource types `oldsnippet` and `oldthumbnail` are both derived from any version of a resource other than the most recently crawled version.

Some usage types – `index`, `preserve`, `present` and its derivatives – may be restricted by qualifiers as specified below and may, if appropriate, be restricted by more than one qualifier.

NOTE – Unless a permitted usage is qualified, no default restrictions can be assumed. For example, a crawler representing a search engine may by default only index a resource until it is re-crawled, but the same is not true of a crawler representing a web archive, which may wish to preserve all versions of a resource indefinitely. It is recommended that all permissions should, where possible, be qualified to indicate any applicable restriction of the usage.

2.5.1 Basic usage type `crawl`

This usage type enables expression of a permission or prohibition to crawl any of a set of one or more resources. There are currently no options for restricting this usage type.

For example:

```
        ACAP-disallow-crawl: /private/
```

NOTE – There is no exact equivalent to this prohibition in conventional REP, because a `Disallow:` field in conventional REP terms may be interpreted by different crawlers to mean either `ACAP-disallow-crawl:` or `ACAP-disallow-index:` in ACAP terms.

2.5.2 Basic usage type `follow`

This usage type enables expression of a permission or prohibition to follow links found within any of a set of one or more crawled resources. There are currently no options for restricting this usage type.

2.5.3 Basic usage type `index`

This usage type enables expression of a permission or prohibition to create index entries for any of a set of one or more crawled resources. Permissions for this usage type may be unrestricted or may be restricted in either or both the ways described below.

2.5.3.1 Permission to index qualified by `time-limit`

A permission to index may be qualified to indicate a time limit restriction indicating for how long this resource may be indexed. A permission of this kind is expressed in the following way:

```
ACAP-allow-index: resource-specification time-limit=value
```

where *`value`* can take any of the following forms:

- `until-recrawled`, indicating that the resource may be indexed until it has been re-indexed following a re-crawl of the same resource.
- `until-`*`yyyy-mm-dd`*, indicating that the resource may be indexed until but not after the specified date *`yyyy-mm-dd`*, which must be a date in the specified ISO date format with hyphens between the year, month and day.
- *`n`*`-days`, indicating that the resource may be indexed from the point in time at which it is first indexed for the specified number of days *n*, which must be an integer value.

Examples of this permission using this qualification:

```
ACAP-allow-index: /public/ time-limit=until-recrawled

ACAP-allow-index: /news/2007/ time-limit=until-2007-12-31

ACAP-allow-index: /current-news/ time-limit=3-days
```

2.5.3.2 Permission to index qualified by `must-use-resource`

● IMPORTANT – This qualification of the `present` usage types could present security or other issues for aggregator users, and will generally only be interpreted by prior arrangement.

A permission to index may be qualified to indicate that the index entries for the crawled resource are not to be derived from the crawled resource in the normal way but from a resource specified using the `must-use-resource` qualifier. A permission of this kind is expressed in the following way:

```
ACAP-allow-index: resource-specification must-use-resource=URI
```

where *`URI`* is either

a regular URI (absolute or relative) that points to a separate resource to be used for indexing purposes, containing the necessary index terms *or*

> a URI belonging to the scheme specified below, identifying a single element within the crawled resource (which must therefore be an HTML page) that is to be used for indexing purposes (i.e. the remainder of the crawled resource may not be indexed).

A single element extracted from an HTML page is identified by:

- *either* the value of an `id` attribute
- or the value of a `class` attribute (in which case only the first matching element is to be extracted)
- *or* the content of a named META tag within the header.

In these cases, the following URI scheme is used:

> `the-acap:extract:id:`*`identifier`*
>
> `the-acap:extract:class:`*`class-name`*
>
> `the-acap:extract:meta:`*`meta-tag-name`*

where *`identifier`* is an identifier on a specific element within an HTML content item, *`class-name`* is the value of a class attribute used to label one or more elements within an HTML content item and *`meta-tag-name`* is the value of the 'name' attribute of a META Tag in the header of the crawled resource.

Examples of permissions to index using the `must-use-resource` qualification:

```
ACAP-allow-index: /page-image.pdf must-use-resource=/page-image-index.txt

ACAP-allow-index: /articles/ must-use-resource=the-acap:extract:class:abstract
```

If a regular URI is specified, the resource specified by this URI is to be used when indexing any resource that matches *`resource-specification`*. The crawler must be permitted to crawl this resource, but other usages of this resource need not be permitted.

NOTE – If each crawled resource needs to be indexed using a different resource, it will generally be more efficient to place such a qualified index permission in a META tag within the resource rather than in a robots.txt file, since otherwise a large number of such permissions may need to be included in the robots.txt file.

NOTE – If the indexing resource is not contained within the crawled resource, there is no guarantee that the indexing resource will be crawled immediately, and in general will be flagged for crawling at a later time. The crawler operator will not be able to index as intended until both resources have been crawled.

2.5.4 Basic usage type `preserve`

This usage type enables expression of a permission or prohibition to preserve a copy of a crawled resource.

2.5.4.1 Permission to `preserve` qualified by `time-limit`

A permission to preserve a copy of a crawled resource may be qualified to indicate a time limit restriction indicating for how long this resource may be preserved. A permission of this kind is expressed in the following way:

```
ACAP-allow-preserve: resource-specification time-limit=value
```

where *`value`* can take any of the following forms:

- `until-recrawled`, indicating that a copy of the resource may be preserve until it has been re-indexed following a re-crawl of the same resource.
- `until-`*`yyyy-mm-dd`*, indicating that a copy of the resource may be preserved until but not after the specified date *`yyyy-mm-dd`*, which must be a date in the specified ISO date format with hyphens between the year, month and day.
- *`n`*`-days`, indicating that a copy of the resource may be preserved from the point in time at which it is first copied for the specified number of days *n*, which must be an integer value.

NOTE – An additional value has been suggested to meet the requirements of web archive use cases to be able to express permission to preserve permanently a crawled resource. But as this does not imply a restriction it is not clear that this should be specified using the `time-limit` qualifier, and so is not ready for implementation in ACAP Version 1.0.

Examples of this permission using this qualification:

```
ACAP-allow-preserve: /public/ time-limit=until-recrawled

ACAP-allow-preserve: /news/2007/ time-limit=until-2007-12-31

ACAP-allow-preserve: /current-news/ time-limit=3-days
```

2.5.5 Basic usage type `present`

The basic usage type `present` enables expression of a general permission or prohibition to present a particular resource in any way. For example, the following unqualified permission would allow any derivation of the resource that matches `/public/` to be presented in any way:

```
ACAP-allow-present: /public/
```

2.5.5.1 Usage types based upon `present`

2.5.5.1.1 `present-original`

This usage type enables expression of a permission or prohibition to present the original of the crawled resource retrieved at the time of presentation from the crawled site, to ensure that the current version of the resource is presented. Technically, this would normally be achieved by embedding the original within a frame. For example, to prohibit this usage entirely:

```
      ACAP-disallow-present-original: /
```

2.5.5.1.2 `present-currentcopy`

This usage type enables expression of a permission or prohibition to present a preserved copy of the most recently crawled version of a resource. It would only be necessary to include an explicit permission or prohibition for this usage type if it is permitted to preserve a copy of the crawled resource.

2.5.5.1.3 `present-oldcopy`

This usage type enables expression of a permission or prohibition to present a preserved copy of a version of a resource other than the most recently crawled version. It would only be necessary to include an explicit permission or prohibition for this usage type if it is permitted to preserve a copy of the crawled resource until after the resource has been re-crawled. This would generally be of relevance only in archival use cases.

2.5.5.1.4 `present-snippet`

This usage type enables expression of a permission or prohibition to present a snippet generated by the aggregator (normally using a proprietary algorithm) for the most recently crawled version of a resource.

2.5.5.1.5 `present-thumbnail`

This usage type enables expression of a permission or prohibition to present a thumbnail image generated by the aggregator of the most recently crawled version of a resource. For example, to allow snippets to be presented but not thumbnails:

```
      ACAP-allow-present-snippet: /public/
      ACAP-disallow-present-thumbnail: /public/
```

2.5.5.1.6 `present-oldsnippet`

This usage type enables expression of a permission or prohibition to present a snippet generated by the aggregator (normally using a proprietary algorithm) for a version of a resource other than the most recently crawled version.

2.5.5.1.7 *`present-oldthumbnail`*

This usage type enables expression of a permission or prohibition to present a thumbnail image generated by the aggregator of a version of a resource other than the most recently crawled version.

2.5.5.1.8 *`present-link`*

This usage type enables expression of a permission or prohibition to present a link to the crawled resource on the crawled site.

2.5.5.2 Qualifications of permitted `present` usage types

A permission field may in certain cases contain one or more qualifiers to restrict the permitted usage. Unless indicated otherwise, any individual qualifier may only occur once in a permission field.

2.5.5.2.1 *`time-limit`*

A time limit for presentation of a resource may be set using the `time-limit` qualifier as specified above for usage types `index` and `preserve`. For example:

```
        ACAP-allow-present: /public/ time-limit=until-recrawled

        ACAP-allow-present-snippet: /news/2007/ time-limit=3-days
```

2.5.5.2.2 *`must-use-resource`*

● IMPORTANT – This qualification of the basic `present` usage type and of its derived usage types is a feature that is not yet fully ready for implementation.

● IMPORTANT – This qualification of the `present` usage types could present security or other issues for search engines, and will generally only be interpreted by prior arrangement.

The `must-use-resource` qualifier enable expression of a permission to present a specified snippet or thumbnail instead of one generated or selected by the aggregator. For example, in the case of a thumbnail, the following permission field could be used to express permission to display a cover image as a thumbnail for any of the pages of a book that appear in a search result:

```
ACAP-allow-present-thumbnail: /book/pages/ must-use-resource=/book/cover.jpg
```

The crawler must be permitted to crawl a resource specified in a `must-use-resource` qualifier, but other usages of this resource need not be permitted.

It is recommended that, if possible, the URI should normally specify a resource within the crawled resource and not external to it, as this is less likely to present technical and security difficulties to the crawler.

2.5.5.2.3 *`max-length`*

A limit on the length of a snippet, in characters or words, may be expressed as follows:

```
ACAP-allow-present-resource-type: resource-specification max-length=value
```

where *`resource-type`* must be any of `snippet` or `oldsnippet` as appropriate and `value` is either *`n`*`-chars` or *`n`*`-words`, with *n* an integer value. For example:

```
        ACAP-allow-present-snippet: /news/ max-length=250-chars
```

NOTE – If a crawler operator is unable to accept restrictions on lengths of snippets, the fall-back interpretation is likely to be to prohibit presentation.

2.5.5.2.4 *`prohibited-modification`*

If a content owner wishes to restrict presentation to exclude certain types of modification, this may be expressed for all kinds of presentation as follows:

```
ACAP-allow-present: r-s prohibited-modification=mod-type
```

or for presentation of specific types of derived resource as follows:

```
ACAP-allow-present-resource-type: r-s prohibited-modification=mod-type
```

where *`resource-type`* may be any of the following:
`original`, `currentcopy`, `oldcopy`;
`r-s` is the resource specification;
and *`mod-type`* may be any of the following:

- `any` – in which case any modification is prohibited
- `format` – in which case a resource format conversion (e.g. from PDF to HTML) is prohibited
- `style` – in which case any typographic style or layout modification is prohibited
- `translation` – in which case any translation of text to another language is prohibited
- `annotation` – in which case any annotation, for example to include end-user ratings or tags, is prohibited.

For example, to prohibit the presentation of end-user annotations in any presentation of a resource:

```
        ACAP-allow-present: /public/ prohibited-modification=annotation
```

To prohibit style changes in presenting a preserved copy:

```
        ACAP-allow-present-currentcopy: /public/ prohibited-modification=style
```

The `prohibited-modification` qualifier may be repeated within a permission, except if the prohibited modification is `any`. For example, to prohibit either format changes or translation when displaying a preserved copy of a resource:

```
ACAP-allow-present-currentcopy: /public/ prohibited-modification=format
prohibited-modification=translation
```

NOTE – The above example would be a single line of text in a robots.txt file.

2.5.5.2.5 *`must-include-resource`*

● IMPORTANT – This qualification of the basic `present` usage type and of its derived usage types is a feature that is not yet fully ready for implementation.

● IMPORTANT – This qualification of the `present` usage types could present security or other issues for search engines, and will generally only be interpreted by prior arrangement.

Permission to present a resource may be made conditional upon inclusion of a particular resource, as follows:

```
        ACAP-allow-present: resource-specification must-include-resource=URI
```

where *`URI`* is either a general URI, in which case the role of resource that must be included is unspecified, or (preferably) a URI constructed according to the following scheme:

`the-acap:`*`resource-role`*`:`*`included-resource-locator`*

where *`resource-role`* can be either of the following:

- `credit` – to indicate that the resource to be included is a credit to be presented with the resource (e.g. with an image)
- ● `link` – to indicate that the resource to be included is to be presented with the link to the original resource (e.g. an additional link to terms and conditions, to a registration page, etc)
  NOTE – the use of `link` is subject to further definition and not yet ready for implementation.

and *included-resource-locator* can be any of the following:

```
id:identifier
class:class-name
meta:meta-tag-name
general-URI
```

and where `id`, `class` and `meta` indicate that the resource that must be included is to be found within the crawled resource (which must therefore be an HTML page), while a *general-URI* would be the URI of a separate resource that must be included.

The crawler must be permitted to crawl a resource specified by a general URI in a `must-include-resource` qualifier, but other usages of this resource need not be permitted.

NOTE – Given that the resource to be included will often be specific to the crawled resource, it is more likely that this qualifier will be used to qualify permissions embedded in META Tags than permissions contained in a robots.txt file. See Part 2 for examples.

It is recommended that the URI should use the scheme described above, to provide information to the crawler on the intended role of the included resource.

It is recommended that, if possible, the URI should normally specify a resource within the crawled resource and not external to it, as this is less likely to present technical and security difficulties to the crawler.

2.5.5.2.6 *prohibited-context*

Permission to present either the original resource or a preserved copy of it may be restricted to prohibit presentation within a frame constructed by the user as follows:

```
ACAP-allow-present-resource-type: r-s prohibited-context=within-user-frame
```

where *resource-type* is one of: `original`, `currentcopy`, `oldcopy`; and *r-s* is the resource specification. For example:

```
ACAP-allow-present-currentcopy: /public/ prohibited-context=within-user-frame
```

2.5.5.2.7 *required-context*

Permission to present either the original resource or a preserved copy of it may be restricted to require presentation within a frame that simulates the way the original resource is presented to an end-user as follows:

```
ACAP-allow-present-resource-type: r-s required-context=within-original-frame
```

where *resource-type* is one of: `original`, `currentcopy`, `oldcopy`; and *r-s* is the resource specification. For example:

```
ACAP-allow-present-original: /public/ required-context=within-original-frame
```

2.5.6 Basic usage type `other`

This usage type enables blanket prohibition of usages other than those that are explicitly permitted, for example:

```
        ACAP-disallow-other: /public/
```

There are currently no options for restricting this usage type.

NOTE – Although in theory the usage type `other` could be used in a permission field, there is no known use case for doing so.

2.6 ACAP action requests

● IMPORTANT – This feature is the subject of continuing discussion and development and is not ready for implementation.

An ACAP action request field contains a request to named or any crawlers to perform a take down or re-crawl action as soon as possible. The following syntax is proposed:

```
        ACAP-request-action: /resource-path
```

where *action* specifies a specific action to be taken by the interpreting system and *resource-path* locates a single resource relative to the server document root. Some crawler operators may interpret this field as only providing confirmation of a request made by other means.

An example of an ACAP action request might be:

```
        ACAP-request-take-down: /news/bad-story.htm
```

2.6.1 Take down

● IMPORTANT – This feature is the subject of continuing discussion and development and is not ready for implementation.

An action request field using the action token `take-down` specifies a request to erase all copies and index entries of the specified resource as soon as possible.

2.6.2 Re-crawl

● IMPORTANT – This feature is the subject of continuing discussion and development and is not ready for implementation.

An action request field using the action token `re-crawl` specifies a request to re-crawl the specified resource as soon as possible.

2.7 ACAP definition fields

A consequence of adopting the ACAP extensions to REP is that robots.txt files will frequently be significantly larger than previously. The larger a file, the more likely it is to contain errors. One way of reducing the size of a file, and thereby reducing the likelihood of errors, is to provide a short-hand for what would otherwise be long forms of expression:

- for complex qualified usage permissions,
- for several usage permissions applying to the same set of resources and
- for complex resource sets requiring multiple resource path patterns to specify them in full.

The method specified here involves the definition of 'macro' names that can be used in permissions to refer to otherwise complex qualified or composite usages, or complex resource sets.

A qualified or composite usage defined in this way is referred to as a locally-defined usage, because it must occur in the same robots.txt file or in one that contains a permission reference to the robots.txt file in which it is used in a permission field.

All ACAP definition fields must be placed before any ACAP records in a robots.txt file. An ACAP definition field must be placed before any other ACAP definition field that relies upon it. Resource set definition fields may occur before or after other definition fields, and their order is immaterial.

NOTE – Each definition field must be a single line of text in a robots.txt file.

2.7.1 Qualified usage definition

A qualified usage definition field defines a qualified usage in terms of a standard ACAP usage and one or more associated usage qualifiers as follows:

```
ACAP-qualified-usage: qualified-usage-name
standard-usage-name qualifiers
```

where *`qualified-usage-name`* is a name assigned to the qualified usage; *`standard-usage-name`* is the name of a standard usage type; and *`qualifiers`* is a sequence of one or more usage qualifier specifications, separated by spaces.

A qualifier specification has following syntax:

```
        qualifier-type=qualifier-value
```

where *`qualifier-type`* is a valid qualifier type for the given usage type and *`qualifier-value`* is a valid value for the specified qualifier type.

For example, if permission to present snippets is always to be restricted by both qualifiers `time-limit` and `max-length`, a qualified usage could be defined as follows:

```
ACAP-qualified-usage:
my-present-snippet present-snippet time-limit=5-days max-length=250-chars
```

then used in a permission field as follows:

```
        ACAP-allow-(my-present-snippet): /public/
```

The parentheses around a locally-defined usage name ensure that it can always be distinguished from a standard usage type name.

2.7.2 Composite usage definition

A composite usage definition defines a usage as a combination of usages, which may be either standard ACAP usage verbs or qualified usages. It has the following proposed syntax:

```
        ACAP-composite-usage: composite-usage-name usage-names
```

where *`composite-usage-name`* is a name assigned to the composite usage and *`usage-names`* is a space-separated list of standard usage types and names of locally-defined qualified usages in parentheses.

A permission field containing a composite usage is to be interpreted as if there were a series of permission fields for each of the constituent usages for the same set of resources.

An example of a composite usage definition might be:

```
ACAP-composite-usage:
my-default-present (present-30-word-snippet) present-thumbnail
```

where `(present-30-word-snippet)` refers to a qualified usage that might be defined (earlier in the robots.txt file) as follows:

```
ACAP-qualified-usage:
present-30-word-snippet present-snippet max-length=30-words
```

Such a composite usage might then subsequently be used in an ACAP permission as follows:

```
        ACAP-allow-(my-default-present): /public/
```

2.7.3 Resource set definition

A resource set definition defines a set of resources that can be referred to by an ACAP permission, prohibition, qualified usage definition or composite usage definition. It has the following proposed syntax:

```
        ACAP-resource-set: set-name resource-path-pattern ...
```

where `resource-path-pattern` is as defined in 2.8 and can be repeated separated by space characters. For example:

```
        ACAP-resource-set: images *.gif *.jpg *.png
```

which could be used in a prohibition field, for example, to prevent images from being indexed as follows:

```
        ACAP-disallow-index: the-acap:resource-set:images
```

2.8 Pattern matching in ACAP fields

The syntax of a conventional robots.txt file is extended to allow pattern matching in resource path patterns and usage purpose patterns using the "wildcard" characters * and $ with the following meanings:

*	represents zero or more characters and may occur anywhere in a pattern
$	indicates that the immediate preceding character must be the last character in any resource path that matches the pattern, and may only occur as the last character in the pattern.

The characters * and $ have the same meaning when they occur in usage purpose patterns within usage purpose fields and in resource path patterns within permission and prohibition fields.

The current syntax of robots.txt, which allows the use of the asterisk character * to mean "any (other) crawler" in a User-agent field, is adopted for use in crawler identification fields.

Pattern matching in ACAP fields should always be case-insensitive.

NOTE – The use of the characters * and $ in resource path patterns is already widely adopted among major crawler operators.

2.9 Relationship between the proposed extensions and the content of existing robots.txt files

If a robots.txt file contains ACAP records, it is also likely to contain conventional records for as long as some crawlers are not capable of interpreting ACAP records. In such cases it is necessary to know whether a crawler capable of interpreting ACAP records is expected also to interpret conventional records.

2.9.1 Ignoring conventional robots.txt fields

By default, a crawler that can interpret ACAP records is expected to interpret both the conventional and ACAP records. However, a field may be included in a robots.txt file to indicate that the conventional records are to be *ignored* by crawlers that can interpret ACAP records, as follows:

```
ACAP-ignore-conventional-records
```

This field should be placed before any ACAP definition fields.

2.9.2 Interpretation of combinations of ACAP and conventional robots.txt fields

When interpreting conventional records in a robots.txt file, ACAP fields within conventional robots.txt records are to be ignored.

When interpreting ACAP records in a robots.txt file, conventional fields within conventional records are to be interpreted as normal *unless* the robots.txt file contains an ACAP directive indicating that conventional records are to be ignored. Conventional fields found within ACAP records are always to be ignored, as their interpretation in such circumstances may be uncertain.

It is strongly recommended that all ACAP records be placed at the end of a robots.txt file, after any conventional records.

If conventional records are not to be ignored and a permission or prohibition field within an ACAP record and corresponding field within a conventional record have conflicting permissions for the same specific pattern of resources, the field within the conventional record is to be ignored.

2.9.3 Comment convention for indicating that a robots.txt file contains ACAP records

A comment may be included at the start of a robots.txt file to indicate that the file contains ACAP records and to indicate the version of the ACAP standard that is supported. This comment should take the following form:

```
##ACAP version=xxx
```

where xxx is the version number, currently 1.0. It is recommended that this comment precede any other comment or record in the file.

3 Formal specification of the syntax of the proposed extensions to the robots.txt format

A formal definition of the syntax of the ACAP extensions to the robots.txt format is given here, using the ABNF notation defined in IETF RFC 2234[5]. “URI” and “relative-part” are defined in IETF RFC 3986[6].

```
ACAP-extensions          = *ignorable [ACAP-ignore-conventional-records]
                           [ACAP-definitions] ACAP-records

ACAP-ignore-conventional-records = “ACAP-ignore-conventional-records”
                           field-end

ACAP-definitions         = 1*ACAP-definition-field

ACAP-records             = 1*ACAP-record

ACAP-record              = (1*named-crawler-identification-field /
                           any-crawler-identification-field) (permissions-
                           reference-field / (permission-fields / ACAP-sub-
                           record) *ACAP-sub-record)

named-crawler-identification-field = “ACAP-crawler:” *WSP crawler-name
                           field-end

any-crawler-identification-field = “ACAP-crawler:” *WSP “*” field-end

crawler-name             = name-start-char *(WSP / VCHAR-excluding-hash)

VCHAR-excluding-hash     = %x21-26 / %x28-7E

permissions-reference-field = “ACAP-permissions-reference:” *WSP
                           (URI / relative-part) field-end

permission-fields        = 1*(basic-permission-field /
                           qualified-permission-field / prohibition-field /
                           action-request-field)

basic-permission-field   = “ACAP-allow-”
                           ((ACAP-usage-name [“-“ used-resource-type-name]) /
                           local-usage-name) “:” *WSP resource-specification
                           field-end
```

```
qualified-permission-field = "ACAP-allow-"
                           ACAP-usage-name ["-" used-resource-type-name] ":"
                           *WSP resource-specification
                           1*(1*WSP qualifier-specification) field-end

prohibition-field        = "ACAP-disallow-"
                           (ACAP-usage-name ["-" used-resource-type-name] ":"
                           *WSP resource-specification field-end
```

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

```
ACAP-usage-name          = name-start-char *name-char
```

NOTE – An `<ACAP-usage-name>` must be the name of a usage defined in the ACAP usage definitions dictionary.

```
local-usage-name         = "(" (qualified-usage-name /
                           composite-usage-name) ")"

used-resource-type-name  = name-start-char *name-char-excluding-hyphen
```

NOTE – A `<used-resource-type-name>` must be the name of a used resource type defined in the ACAP usage definitions dictionary, and the combination of ACAP usage name and used resource type must be a valid combination defined in the dictionary.

```
resource-specification   = resource-path-pattern /
                           ("the-acap:resource-set:" resource-set-name)

resource-path-pattern    = relative-part
```

NOTE – A `<resource-path-pattern>` may contain the reserved delimiter characters "*" and "$" (see Section 2.2 of IETF RFC 3986) that are to be interpreted as defined above. Other uses of `<relative-part>` may not contain these reserved characters.

```
action-request-field     = "ACAP-request-" action-name ":" *WSP relative-part
                           field-end

action-name              = "take-down" / "re-crawl"

ACAP-sub-record          = 1*usage-purpose-pattern-field
                           1*(permission-field / prohibition-field)

usage-purpose-pattern-field = "ACAP-usage-purpose:" *WSP (name /
                           URI-pattern) field-end
```

NOTE – A `<URI-pattern>` has the same syntax as a `<URI>`, except that it may not contain the character "#", as this would be interpreted as the start of a comment, and it may contain "*", to be interpreted as a wildcard character.

```
ACAP-definition-field    = qualified-usage-definition-field /
                           composite-usage-definition-field /
                           resource-set-definition-field

qualified-usage-definition-field = “ACAP-qualified-usage:” *WSP
                           qualified-usage-name 1*WSP ACAP-usage-name
                           [“-“ used-resource-type-name]
                           1*(1*WSP qualifier-specification) field-end

qualified-usage-name     = name-start-char *name-char

qualifier-specification  = qualifier-type *WSP “=” *WSP
                           qualifier-value

qualifier-type           = name-start-char *name-char

qualifier-value          = 1*value-char

value-char               = %x21 / %x24-26 / %x28-7E
```

NOTE – A <`qualifier-value`> may not contain the character #, as this would be interpreted as the start of a comment. The quotation mark “ and apostrophe ‘ are also excluded, to avoid ambiguity of whether or not they form part of the value. All three characters may be represented by percent character notation %nn, as may the percent character itself. Unicode characters outside the normal ASCII range, i.e. above hexadecimal value 7E (126) may be represented by UTF-8 characters in percent character notation..

```
composite-usage-definition-field = “ACAP-composite-usage:” *WSP
                           composite-usage-name 1*(1*WSP (ACAP-usage-name
                           [“-“ used-resource-type-name] /
                           local-usage-name)) field-end
```

NOTE – A <`local-usage-name`> in a <`composite-usage-definition`> must correspond to an <`qualified-usage-name`> defined in a <`qualified-usage-definition`> in the same permissions resource.

```
composite-usage-name     = name-start-char *name-char
```

NOTE – No two qualified or composite usages may share the same name.

```
resource-set-definition-field = “ACAP-resource-set:” *WSP resource-set-name
                           1*(1*WSP resource-path-pattern) field-end

resource-set-name        = name-start-char *name-char
```

NOTE – All names are case-insensitive, so upper- and lowercase forms of the same letter should be treated as equivalent.

```
name-start-char            = ALPHA

name-char                  = ALPHA / DIGIT / “-“ / “_” / “.”

name-char-excluding-hyphen = ALPHA / DIGIT / “_” / “.”

field-end                  = 1*ignorable

ignorable                  = blank-field / comment-field

blank-field                = *WSP CRLF

comment-field              = *WSP comment CRLF

comment                    = “#” *(WSP / VCHAR)
```

4 Outstanding issues not covered in this version of the ACAP extensions to REP

A number of outstanding issues remain to be resolved in future versions of the ACAP proposed extensions to the Robots Exclusion Protocol. These include:

- clarification of the meaning of certain qualifiers with specific usage types, especially `must-use-resource` with usage type `present` and its derivatives
- specification of further usage types and qualifier types and values to meet the requirements of additional use cases
- decision as to whether all qualifiers must be specified as restrictions on permissions, or can include positive statements that there is no restriction on certain usages that might by default be considered to be restricted (e.g. to specify that a resource may be preserved indefinitely in web archival use cases)
- specification of recommended limits on various metrics that relate to the size and complexity of a robots.txt file, including but not necessarily limited to:
  – maximum file size in bytes,
  – maximum field length in characters,
  – maximum number of records,
  – maximum number of sub-records per record,
  – maximum number of fields per record,
  – maximum number of fields per sub-record.

5 References

1. Robots Exclusion Protocol: An informal specification, based upon an original June 1994 “consensus” of robot authors and others, can be found on the web at http://www.robotstxt.org/, including guidance on both the robots.txt format and the use of Robots META Tags. A number of extensions have been proposed by major search engine operators and others, and some of these extensions are in widespread use.

2. ACAP Technical Framework – Communicating access and usage policies to crawlers using extensions to the Robots Exclusion Protocol – Part 2: Extension of the Robots META Tag format and other techniques for embedding permissions in HTML content. Current version available at http://www.the-acap.org/download.php?ACAP-TF-CrawlerCommunications-Part2-V1.0.pdf.

3. ACAP Dictionary of Access and Usage Terminology. Current version available at http://www.the-acap.org/download.php?ACAP-TF-Dictionary-V1.0.pdf.

4. Hypertext Transfer Protocol – HTTP/1.1 (Internet RFC 2616), Internet Engineering Task Force, June 1999 – see http://www.ietf.org/rfc/rfc2616.txt for more details

5. Augmented BNF for Syntax Specifications: ABNF (Internet RFC 2234), Internet Engineering Task Force, November 1997 – see http://www.ietf.org/rfc/rfc2234.txt for more details.

6. Uniform Resource Identifier (URI): Generic Syntax (Internet RFC 3986) , Internet Engineering Task Force, January 2005 – see http://www.ietf.org/rfc/rfc3986.txt for more details.