

Strawman proposals for extension of the Robots Exclusion Protocol Part 1: Extension of robots.txt format

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 (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. The method of communication is in two parts: a format for a file containing permissions directives to crawlers, called 'robots.txt'; and a format for embedding permissions in HTML page headers, called Robots META Tags.

Despite its widespread use, REP is not a formal standard and is indeed interpreted in various, inconsistent ways by different web crawler operators. In order for ACAP to be useful, publishers need to be certain that permissions will be interpreted consistently by crawler operators. The major web crawler operators, including the major search engine operators, have indicated that they see no practical alternative to REP for communicating permissions for the foreseeable future.

This document is the first 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.

1 What is REP?

The closest to a single specification of REP, as currently used, is to be found in a series of web documents¹, which include descriptions of both 'robots.txt' and Robots META Tags (the subject of Part 2 of these proposals).

This document deals with the first part of REP, namely the 'robots.txt' file format.

The web documents that describe 'robots.txt' are:

- the June 1994 protocol description, which represents a consensus that was reached among crawler developers at that time on a method of communication of permissions using a resource named 'robots.txt', and a format for the permissions contained in 'robots.txt'

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

- a draft Internet Specification (draft RFC), dated November 1996, that has never been ratified or implemented, and which attempts to formalise and extend the format defined in the 1994 document.

In order to understand how 'robots.txt' is interpreted by a web crawler one must also refer to the guidance documents and FAQs provided by the individual crawler operators on their websites, and they are as much part of what REP is as are the original documents. They specify a number of extensions to the original REP specifications, some of which have become widely adopted among at least the major search engines.

REP, despite being informally defined, has become widely used due to its adoption by the major search engines, who have encouraged the webmasters of millions of websites to use it. The major search engines, having adopted REP (and thereby ensuring its survival) and being the main recipients and interpreters of REP, are its *de facto* guardians.

The major search engines have been talking for some time about the possibility of standardising REP, but there are as yet no public proposals. In the meantime new extensions are being developed and supported by one or more search engines but only a few extensions are widely adopted.

2 Can ACAP use 'robots.txt' as it stands?

Fundamentally, ACAP requires consistent and unambiguous interpretation of all its permissions. As it stands, the specification of 'robots.txt' is insufficiently clear to be consistently interpreted, and, if one excludes a number of proprietary extensions, only supports very limited forms of permissions communication.

An example of the inadequacy of REP is that the format is unclear about how to interpret overlapping permissions. This inevitably results in inconsistent interpretation. It is understood that some crawlers look for the first permissions that apply to them and to the resources they wish to retrieve, and ignore subsequent permissions; while other crawlers look for the permission that most closely matches the resource they wish to retrieve, wherever it may be in 'robots.txt'.

ACAP also needs to be able to communicate much more than whether or not a crawler may crawl a particular set of resources. More can be achieved using Robots META tags, but ACAP needs to communicate site-level permissions as well as permissions associated with individual resources.

3 What can be done to change 'robots.txt'?

In the recent past the development of REP has been effectively controlled by its major recipients and guardians, the major search engines. We believe that it will assist the guardians of REP if ACAP makes concrete proposals as to how REP might

be changed to enable ACAP permissions to be communicated fully and unambiguously.

Given the huge established base of websites that use REP, it would be impractical to propose to make changes to the existing 'robots.txt' format. The safest approach is to define an extension of the format that existing crawlers will ignore until such time as they are re-programmed to interpret the ACAP permissions.

3.1 A possible approach to extension of the 'robots.txt' format

The current rule in REP is that any line beginning with a hash symbol '#' is treated by crawlers as a human-readable comment. One possible approach to the extension of REP would therefore be to define a special class of such comments that are very unlikely to be found in existing 'robots.txt' files.

For example, a line in 'robots.txt' prefixed by *two* hash symbols immediately followed by the letters A, C, A and P would currently be treated as a comment and would be extremely unlikely ever to have been used in existing 'robots.txt' files, viz:

```
##ACAP ...
```

Programming a crawler to recognise some comments as significant data would add complexity and risk to the process of parsing a robots.txt file, in which it has hitherto been possible to ignore all comments.

An alternative would be to use an agreed extension mechanism that is acceptable to search engine operators, and that would in the same way be ignored by those search systems that are not specifically programmed to recognise ACAP permissions. In this case an ACAP permission could simply be recognised by having the initial token

```
ACAP-
```

at the start of each line in 'robots.txt' that contains an ACAP permission field.

Following consultation with crawler operators, it has been decided to propose the latter approach, as the one that creates least risk for crawlers not programmed to recognise ACAP permission fields.

The question of what will follow `ACAP-` depends upon the ACAP standard usages that need to be expressed. However, there is an important question that has to be resolved with the existing format: whether or not to use the existing mechanism within REP for associating permissions with a specific crawler.

The existing mechanism allows crawlers to be identified individually by name, e.g.

```
User-Agent: crawlername  
Disallow: ...
```

or alternatively identified by default, using the asterisk * to mean "all except those named above/below", e.g.

```
User-Agent: *  
Disallow: ...
```

Unfortunately, crawler operators are not consistent in their interpretation of overlapping sets of permissions. ACAP might therefore propose a consistent interpretation along the following lines:

- permissions associated with more specific (narrower) sets of resources override permissions associated with less specific (broader) sets of resources
- permissions associated with a specific resource that are addressed to a specific crawler override default crawler permissions associated with the same resource.

If a consistent interpretation could be agreed, there would be no reason for not using the existing mechanism within REP for identifying whether permissions are addressed to specific named crawlers or to crawlers in general. However, such agreement is not certain and in any case is likely to take some time to reach, so in the meantime an alternative mechanism for addressing crawlers must be included in these strawman proposals.

4 Strawman proposals to extend the 'robots.txt' format

The following are "strawman proposals" for extension of the robots.txt file format, as defined as part of the Robots Exclusion Protocol. These proposals are entirely provisional and subject to change in the light of test results.

4.1 General

ACAP permissions in 'robots.txt' are expressible in one or more ACAP records. An ACAP record defines a set of permissions, prohibitions and 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 comments².

An ACAP record may, if appropriate, contain sub-records that group permission and prohibition fields according to 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

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.

which case the general permissions and prohibitions occur before the sub-records in the ACAP record.

In order to avoid ambiguity and overlap, the following rules are proposed. For the purposes of processing conventional robots.txt records, ACAP fields within conventional robots.txt records are to be ignored. For the purposes of processing ACAP records, conventional fields within ACAP records are to be interpreted as normal *unless* the robots.txt file contains an ACAP directive indicating that conventional records are to be ignored. It will probably prove to be good practice to place all ACAP records at the end of a robots.txt file, after any conventional records.

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, unless the robots.txt file contains a directive that they should be ignored, all conventional records must be interpreted by a crawler to determine which to apply to any given resource. In cases where fields within a record have contradictory or overlapping interpretations, a mechanism for resolving such conflicts is proposed below.

The interpretation of a field may depend upon interpretation of the definition of an ACAP qualified usage or composite usage. ACAP qualified usage and composite usage definitions must precede the records containing fields that use them.

An ACAP record may either be addressed to one or more named crawlers or to “any crawler”. Permissions and prohibitions contained in an ACAP record addressed to one or more named crawlers override permissions with the same usage purposes (if any) and usages and matching the same resources contained in an ACAP record addressed to “any crawler”.

In order to keep the syntax of ACAP records as simple as possible, it is proposed that complex qualifications and combinations of usages be defined separately in the robots.txt file, and referred to simply by name in ACAP records. A special syntax for defining complex usages – referred to as “locally-defined usages” – is proposed.

A special syntax is also proposed for including references to permissions held in another resource.

4.2 Addressing ACAP permissions to crawlers

An ACAP record begins with one or more fields that specify the crawlers to which the record is addressed. An ACAP record may be addressed *either* to one or more identified crawlers *or* to all crawlers. Only one record may be addressed to all crawlers.

All ACAP records that specify a crawler by name are addressed to that crawler, as is any ACAP record that is addressed to all crawlers. It is expected that a crawler will merge all the ACAP records that specify that crawler by name and selectively merge

any ACAP record addressed to all crawlers, ignoring permissions and prohibitions that are overridden (see above).

An ACAP record may optionally specify the usage purposes for which the permissions are being communicated, in which case the record is divided into sub-records, each specifying the usage purposes to which the permissions in that sub-record relate.

4.3 Resolving conflicts between overlapping permissions and prohibitions

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

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

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.

See 4.5.2.5.3 for more information on how the permission or prohibition with narrowest scope is to be determined.

4.4 Default interpretations of robots.txt records

If the robots.txt file is silent on whether a particular usage is permitted or not for a specific resource – i.e. the URI of the resource matches no resource path pattern for the specified usage – the usage is permitted.

4.5 Syntax of ACAP extensions to the robots.txt format

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

4.5.1 Pattern matching in ACAP fields

It is proposed that the syntax of a conventional robots.txt file be extended to allow pattern matching in resource path 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.

ACAP proposes that the characters * and \$ have the same meaning when they occur in service locator patterns and resource path patterns within ACAP 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.

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

4.5.2 ACAP record

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.

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

4.5.2.1 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 to make it clear whether a record is an ACAP record or a conventional record. At this stage it is felt necessary to keep the two kinds of record syntactically distinct, so that both may be included in a single robots.txt file.

A crawler identification field has the following proposed syntax:

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

where *crawler-name* is either the name recognised by a specific crawler or an asterisk *, which is interpreted to mean “any crawler”.

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

```
ACAP-crawler: *
```

and may not contain any other crawler identification fields.

4.5.2.2 ACAP permission reference definition field

An ACAP permission reference definition links to the resource containing ACAP permissions in standard XML syntax. The following syntax is proposed:

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

where *resource-locator* is either a path relative to the server document root for a resource located on the same server or a URI for a resource located on another server.

4.5.2.3 ACAP action request field

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, i.e. *take-down* or *re-crawl*, 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
```

4.5.2.4 Sub-record

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 ends 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.

4.5.2.4.1 Usage purpose pattern field

A sub-record begins with one or more usage purpose pattern fields. A usage purpose pattern field has the following proposed 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 a URI that specifies a service or process operated by the operator of a crawler to which the record as a whole is addressed and supplied with input resources by the identified crawler.

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

```
ACAP-usage-purpose: http://service.aggregator.*/*
```

4.5.2.5 Fields used in both ACAP records and sub-records

4.5.2.5.1 ACAP permission field

An ACAP permission field defines a permitted usage for a specified set of resources. In its basic form it has the following proposed syntax alternatives:

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

where *usage* is a standard ACAP usage verb (see 4.5.3.1 for a table of standard ACAP usage verbs);

or, in the case of a usage verb that can be used in conjunction with standard used resource types:

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

where the following are currently the only valid combinations of *usage* and *resourcetype*:

```
present-original  
present-currentcopy  
present-oldcopy  
present-snippet3  
present-extract  
present-thumbnail  
present-oldsnippet  
present-oldextract  
present-oldthumbnail  
present-link
```

or the following proposed syntax:

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

where *usage* between parentheses is the name of a locally-defined usage that is defined earlier in the same 'robots.txt' file;

and *resource-specification* is either

a resource path pattern that matches one or more paths to resources relative to the server document root /

or

the name of a resource set defined elsewhere in the same robots.txt file prefixed by `acap:resource-set:` (see 4.5.3.3).

It is proposed that this syntax be extendible by adding qualifiers after the resource specification. The syntax for qualifiers is introduced in section 4.5.3.1 and specified more formally in section 4.5.6.

An example of an ACAP permission using a standard ACAP usage verb might be:

```
ACAP-allow-index: /public/*.htm
```

or in the case of a locally-defined resource set called `stories`:

```
ACAP-allow-index: acap:resource-set:stories
```

³ The tokens `present-snippet`, `present-extract` and `present-thumbnail` all are with reference to the current (i.e. most recently crawled) version of a resource. The tokens `present-oldsnippet`, `present-oldextract` and `present-oldthumbnail` all are with reference to an old (i.e. *not* the most recently crawled) version of a resource.

An example of an ACAP permission using a standard usage verb with associated used resource type might be:

```
ACAP-allow-present-snippet: /public/*.htm
```

An example of an ACAP permission using a locally-defined usage verb might be:

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

4.5.2.5.2 ACAP prohibition field

An ACAP prohibition field defines a prohibited usage for a specified set of resources. It has the following proposed syntax alternatives:

```
ACAP-disallow-usage: resource-path-pattern
```

or

```
ACAP-disallow-usage-resource-type: resource-path-pattern
```

where *usage* and *resource-type* are as defined in the previous section.

An ACAP prohibition may not refer to qualified or composite usages.

An example of an ACAP prohibition using a standard ACAP usage verb might be:

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

4.5.2.5.3 Resolving conflicts between permission and prohibition fields that are matched by the same resource

As discussed above in 4.3, the resolution of conflicting permissions relies upon determining the permission with the narrowest scope. Determining which permission has the narrowest scope depends upon a comparison of the resource path patterns of the conflicting permission fields.

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.

UNRESOLVED ISSUE – This method will not always be reliable, for example when a pattern that contains no wildcard character happens to match a directory/folder name. This method will be tested and, if found to be insufficiently reliable, an alternative will method will be found.

4.5.3 Locally-defined usages and resource sets

Locally-defined usages are of two types: qualified usages and composite usages. To avoid circular definitions, qualified usages may only be defined in terms of standard ACAP usages and usage qualifiers, and composite usages may only be defined in terms of standard ACAP usages and locally-defined qualified usages.

All locally-defined usages must be defined in the robots.txt file in which they are used in permission and prohibition fields, and they must be defined before the ACAP records in which they are used.

Resource sets are sets of resources that require multiple resource path patterns to specify them fully, to which a locally-defined name can be assigned. This name can then be used to enable a single permission or prohibition to be applied to the whole set.

4.5.3.1 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. It has the following proposed syntax:

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

The token *qualified-usage-name* is a name assigned to the qualified usage.

The token *standard-usage-name* is the name of a standard ACAP usage verb, which may be followed by a standard used resource type in appropriate cases (see section 4.5.2.5.1), and *qualifiers* is a sequence of one or more usage qualifier specifications, separated by spaces. A qualifier specification is proposed to have following syntax:

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

where *qualifier-type* is a standard ACAP usage qualifier type and *qualifier-value* is a value in the value domain for the specified qualifier type.

An example of a qualified usage definition might be (displayed over two lines here, but in robots.txt it would occupy only a single line):

```
ACAP-qualified-usage: present-in-original-frame  
present-original required-context=within-original-frame
```

The standard usages and usage qualifiers described in the table below may be used in defining qualified usages. The descriptions provided in the final column are not definitive – for formal definitions of qualifiers and their permitted values see the ACAP usage definitions dictionary.

GUIDANCE ON READING THE TABLE

In the descriptions of each qualifier and value the term “permission” is a short-hand for “permission or prohibition”.

Under the qualifier values column terms in italics represent variable values (e.g. a data or a URI) whereas terms in quotation marks are fixed code values (the quotation marks are not part of the code value and should be omitted when using the code value in an ACAP permission/prohibition).



Usage name	Qualifier type	Usage qualifier value	Description
crawl	none	not applicable	
follow	none	not applicable	
index	must-use-resource	URI	if permission only applies to indexing the fragments of the resource or alternative content identified by the specified URI
index	time-limit	“until-recrawled”	if permission only applies until the next time the resource is crawled
		“until-YYYY-MM-DD”	if permission only applies until the specified date
		“n-days”	if permission only applies until <i>n</i> days after the resource was first placed in the index
preserve	time-limit	“until-recrawled”	if permission only applies until the next time the resource is crawled
		“until-YYYY-MM-DD”	if permission only applies until the specified date
		“n-days”	if permission only applies until <i>n</i> days after the resource was first preserved
present	must-use-resource <i>NOTE – This syntax does not apply to standard used resource types</i>	URI	if permission applies to presentation of a fragment of the resource or alternative content identified by the specified URI

Usage name	Qualifier type	Usage qualifier value	Description
present-snippet	max-length	"n-chars"	length limit in characters
present-extract		"n-words"	length limit in words
present-oldsnippet			
present-oldextract			
present	prohibited-modification <i>NOTE – This qualifier may be repeated, except if the value "any" is included</i>	"any"	if permission only applies to presentation of a resource without modification of any kind
present-original		"format"	if permission only applies to presentation of a resource without modifications to the data format
present-currentcopy		"style"	If permission only applies to presentation of a resource without modifications to the typographic or layout style
present-oldcopy		"translation"	If permission only applies to presentation of a resource without translation of its text content from one written language to another
		"annotation"	if permission only applies to presentation of a resource without annotation of the content with any material (text, symbols, media objects) not derived from the content itself
present	must-include-resource	URI	if permission only applies to presentation of a resource that includes the resource at the specified URI – used to indicate that a specific resource must be included when presenting the used resource
present	time-limit	"until-recrawled"	if permission only applies until the next time the resource is crawled
		"until-YYYY-MM-DD"	if permission only applies until the specified date
		"n-days"	if permission only applies until n days after the resource was first available to be presented



Usage name	Qualifier type	Usage qualifier value	Description
present-original present-currentcopy present-oldcopy	prohibited-context	"within-user-frame"	if permission only applies to presentation of the original resource or a preserved copy, provided it is not embedded in a user-generated presentation context or frame
present-original present-currentcopy present-oldcopy	required-context	"within-original-frame"	if permission only applies to presentation of the original resource or a preserved copy, provided it is embedded in its original context or frame
other	none	not applicable	

4.5.3.1.1 Used resource types in qualified index and present usages

If the qualifier `used-resource` is used in a qualified usage and the value is a URI, the URI may be *either*

a regular URI (absolute or relative) that points to a separate resource to be used for indexing or presentation purposes

or

a URI belonging to the ACAP URI scheme, identifying the resource type as well as the resource location.

If the usage is `index`, it is proposed that the resource used for indexing may be:

- *either* the entire retrieved content item;
- *or* a separate resource containing the necessary index terms;
- *or*, if the retrieved content item is an HTML page, a single element extracted from the page.

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 latter cases, the following uses of the ACAP URI scheme are proposed:

```
acap:extract:id:identifier
```

```
acap:extract:class:class-name
```

```
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 name of a META Tag in the header of the HTML content item whose content is to be used.

If the usage is `present`, it is proposed that the resource used for presentation may be:

- *either* unspecified, implying the presentation of any resource type;
- *or* a separate resource;
- *or*, if the retrieved content item is an HTML page, a single element extracted from the page.

A single element extracted from an HTML page is identified by a URI using the proposed ACAP URI scheme described above.

4.5.3.1.2 *Included resource types in qualified present usages*

If the qualifier `must-include-resource` is used in a qualified usage based on `present`, the URI specifies a resource to be included when presenting the used resource. This URI can be a general URI, in which case the type of resource to be included is unspecified, or it can be a URI in the ACAP scheme, in which case the type of resource is indicated as follows:

```
acap:resource-type:included-resource-locator
```

where `resource-type` can be any one of the following:

```
credit link icon text
```

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 stored content item (which must therefore be an HTML page), while a `general-URI` would be indicate that the resource that must be included is a separate resource.

4.5.3.2 **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 by which it is referred to in ACAP permission fields and `usage-names` is a space-separated list of standard ACAP usage verbs and local names of defined qualified usages.

An ACAP permission field containing a composite usage is to be interpreted as if there were a series of ACAP 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: display-in-frame
  (display-original) (display-copy-in-frame)
```

where `(display-copy-in-frame)` refers to a qualified usage that might be defined as follows:

```
ACAP-qualified-usage: display-copy-in-frame
  present used-resource=stored-copy context=within-frame
```

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

```
ACAP-disallow-(display-in-frame): /
```

Note that the delimiters “(“ and “)” are used when referring to a locally-defined qualified or composite usage in composite definitions and in permissions, but the delimiters are omitted when defining a usage.

4.5.3.3 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 above in 4.5.2.5.1 and can be repeated.

4.5.4 Comment convention for indicating ACAP version

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

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

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

4.5.5 ACAP directive to ignore conventional records

A robots.txt file may contain both conventional and ACAP records. By default, a crawler that can interpret ACAP records is expected to interpret both the conventional and ACAP records. However, it is proposed that a single-line directive 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. The proposed syntax for this directive is:

ACAP-ignore-conventional-records

4.5.6 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.

```

ACAP-proposed-extension = *ignorable [ACAP-ignore-conventional-records]
                           [ACAP-local-definitions] ACAP-records

ACAP-ignore-conventional-records = "ACAP-ignore-conventional-records"
                                   field-end

ACAP-local-definitions = 1*ACAP-local-definition

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 *name-char

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

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

permission-field = "ACAP-allow-"
                  ((ACAP-usage-name ["-" used-resource-type-name]) /
                   local-usage-name) ":" *WSP resource-specification
                  *(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 /
                          ("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-local-definition  = 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*VCHAR
```

NOTE – A <qualifier-value> may not contain the character "#", as this would be interpreted as the start of a comment. Unicode characters outside the normal ASCII range, i.e. above hexadecimal value 7E (126) may be represented by numeric value notation %xnnnn..

```
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               = *ignorable
```

```
ignorable               = blank-field / comment-field
```

```
blank-field             = *WSP CRLF
```

```
comment-field           = *WSP comment CRLF
```

```
comment                 = "#" *(WSP / VCHAR)
```

4.6 Recommended limits on the size of a 'robots.txt' file

It is proposed that there should be recommended limits on the size of a 'robots.txt' file. Recommended limits should be defined in terms of the following parameters:

- overall length of file in bytes
- length of field
- number of records per file
- number of sub-records per record
- number of fields per sub-record
- number of fields per record.

4.7 Outline of a complete ACAP section in a robots.txt file

```
##ACAP version=0.2 # pilot version
# The following section of this robots.txt file is
# addressed to crawlers that are able to read ACAP permissions

# Locally-defined usages
ACAP-qualified-usage: short-term-preserve preserve time-limit=30-days
ACAP-qualified-usage: present-original-format present prohibited-
modification=format
ACAP-resource-set: foobar /public/foo /public/bar

# Permissions for crawlers in general
ACAP-crawler: *
ACAP-usage-purpose: *
ACAP-allow-crawl: /
ACAP-allow-index: /public
ACAP-disallow-index: /
ACAP-allow-preserve: /public
ACAP-allow-(short-term-preserve): /news
ACAP-disallow-preserve: /
ACAP-allow-(present-original-format): *.pdf
ACAP-allow-present: /public
ACAP-disallow-present-thumbnail: /public
ACAP-disallow-present: /
ACAP-disallow-present: acap:resource-set:foobar

# Permissions for crawler xyz
ACAP-crawler: xyz
# Full ACAP permissions can be found in a robots.txt file
# at the following location.
ACAP-permissions-reference: /xyz/robots.txt
```