Web Annotation Protocol

W3C Recommendation 23 February 2017

This version:
https://www.w3.org/TR/2017/REC-annotation-protocol-20170223/

Latest published version:
https://www.w3.org/TR/annotation-protocol/

Latest editor's draft:
https://w3c.github.io/web-annotation/protocol/wd/

Implementation report:
https://w3c.github.io/test-results/annotation-protocol/all.html

Previous version:
https://www.w3.org/TR/2017/PR-annotation-protocol-20170117/

Editor:
Robert Sanderson, J Paul Getty Trust, rsanderson@getty.edu,

Repository:
Github Repository

Changes:
Diff to previous version
Commit history

Please check the **errata** for any errors or issues reported since publication.

This document is also available in this non-normative format: ePub

The English version of this specification is the only normative version. Non-normative translations may also be available.

Abstract

Annotations are typically used to convey information about a resource or associations between resources. Simple examples include a comment or tag on a single web page or image, or a blog post about a news article.

The Web Annotation Protocol describes the transport mechanisms for creating and managing annotations in a method that is consistent with the Web Architecture and REST best practices.

Status of This Document

This section describes the status of this document at the time of its publication. Other documents may supersede this document. A list of current W3C publications and the latest revision of this technical report can be found in the W3C technical reports index at https://www.w3.org/TR/.

By publishing this Recommendation, W3C expects that the functionality specified in this Recommendation will not be affected by changes to the Activity Streams 2.0 [activitystreams-core] and Activity Vocabulary [activitystreams-vocabulary] as those specifications proceed to Recommendation.

This document was published by the Web Annotation Working Group as a Recommendation. If you wish to make comments regarding this document, please send them to public-annotation@w3.org (subscribe, archives). All comments are welcome.

Please see the Working Group's implementation report.

This document has been reviewed by W3C Members, by software developers, and by other W3C groups and interested parties, and is endorsed by the Director as a W3C Recommendation. It is a stable document and may be used as reference material or cited from another document. W3C's role in making the Recommendation is to draw attention to the specification and to promote its widespread deployment. This enhances the functionality and interoperability of the Web.

This document was produced by a group operating under the 5 February 2004 W3C Patent Policy. W3C maintains a public list of any patent disclosures made in connection with the deliverables of the group; that page also includes instructions for disclosing a patent. An individual who has actual knowledge of a patent which the individual believes contains Essential Claim(s) must disclose the information in accordance with section 6 of the W3C Patent Policy.

This document is governed by the 1 September 2015 W3C Process Document.

Table of Contents

1. Introduction

This section is non-normative.

Interoperability between systems has two basic aspects: the syntax and semantics of the data that is moved between the systems, and the transport mechanism for that movement. The HTTP protocol and the Web architecture provides us with a great starting point for a standardized transport layer, and can be used to move content between systems easily and effectively. Building upon these

foundations allows us to make use of existing technology and patterns to ensure consistency and ease of development.

The Web Annotation Protocol describes a transport mechanism for creating, managing, and retrieving Annotations. Annotations in this specification are assumed to follow the requirements of the Web Annotation Data Model [annotation-model] and Web Annotation Vocabulary [annotation-vocab]. This specification builds upon REST principles and the Linked Data Platform [ldp] recommendation, and familiarity with it is recommended.

1.1 Aims of the Protocol

The primary aim of the Web Annotation Protocol is to provide a standard set of interactions that allow annotation clients and servers to interoperate seamlessly. By being able to discover annotation protocol end-points and how to interact with them, clients can be configured either automatically or by the user to store annotations in any compatible remote system, rather than being locked in to a single client and server pair.

1.2 Summary

For those familiar with the Web Annotation model, LDP, and REST, much of the Annotation Protocol will be very obvious. The following aspects are the most important new requirements.

- The media type to use for Annotations is: `application/ld+json;profile="http://www.w3.org/ns/anno.jsonld"`
- Annotation Containers are constrained by the set of constraints described in this specification, and thus the `ldp:constrainedBy` URL is `http://www.w3.org/TR/annotation-protocol/`
- The link header can refer from any resource to an Annotation Container using a `rel` type of: `http://www.w3.org/ns/oa#annotationService`
- The response from a Container after creating an Annotation should include a representation of the Annotation, after any changes have been made to it, in the JSON-LD serialization.
- Annotation Containers should only contain Annotations, and not other resources.
- Activity Streams Collection [activitystreams-core] model is used for paging, as in-page ordering is an important requirement.

1.3 Conformance

As well as sections marked as non-normative, all authoring guidelines, diagrams, examples, and

notes in this specification are non-normative. Everything else in this specification is normative.

The key words may, must, must not, recommended, should, and should not are to be interpreted as described in [RFC2119].

1.4 Terminology

IRI
: An IRI, or Internationalized Resource Identifier, is an extension to the URI specification to allow characters from Unicode, whereas URIs must be made up of a subset of ASCII characters. There is a mapping algorithm for translating between IRIs and the equivalent encoded URI form. IRIs are defined by [rfc3987].

Resource
: An item of interest that may be identified by an IRI.

Web Server
: A program that accepts connections in order to service HTTP requests by sending HTTP responses.

Annotation
: A web resource that follows the Web Annotation Data Model [annotation-model].

Annotation Server
: A Web Server that also makes available and allows the management of Annotations via the protocol described in this document.

Annotation Client
: A program that establishes connections to Annotation Servers for the purpose of retrieving and managing Annotations via the protocol described in this document.

Annotation Container
: An LDP Container [ldp] used to manage Annotations.

2. Web Annotation Protocol Principles

The Web Annotation Protocol is defined using the following basic principles:

- The protocol is developed within the framework laid out by the Web Architecture.
- Interactions will follow the REST best practice guidelines when there is a resource being acted upon.
- Interactions are designed to take place over HTTP.
- Existing specifications and systems will be re-used whenever possible, constrained further when necessary, with invention of new specifications only as a last resort.
- Simplicity and ease of implementation are important design criteria, but ultimately subjective

and less important than the above principles.

3. Annotation Retrieval

The Annotation Server must support the following HTTP methods on the Annotation's IRI:

- `GET` (retrieve the description of the Annotation),
- `HEAD` (retrieve the headers of the Annotation without an entity-body),
- `OPTIONS` (enable CORS pre-flight requests [cors]).

Servers should use HTTPS rather than HTTP for all interactions, including retrieval of Annotations.

Servers must support the JSON-LD representation using the Web Annotation profile. These responses must have a `Content-Type` header with the `application/ld+json` media type, and it should have the Web Annotation profile IRI of `http://www.w3.org/ns/anno.jsonld` in the `profile` parameter.

Servers should support a Turtle representation, and may support other formats. If more than one representation of the Annotation is available, then the server should support content negotiation. Content negotiation for different serializations is performed by including the desired media type in the HTTP `Accept` header of the request, however clients cannot assume that the server will honor their preferences [rfc7231].

Servers may support different JSON-LD profiles. Content negotiation for different JSON-LD profiles is performed by adding a `profile` parameter to the JSON-LD media type in a space separated, quoted list as part of the `Accept` header.

Servers should use the 200 HTTP status code when no errors occurred while processing the request to retrieve an Annotation, and may use 3XX HTTP status codes to redirect to a new location.

The response from the Annotation Server must have a `Link` header entry where the target IRI is `http://www.w3.org/ns/ldp#Resource` and the `rel` parameter value is `type`. The Annotation type of `http://www.w3.org/ns/oa#Annotation` may also be added with the same `rel` type. This is to let client systems know that the retrieved representation is a Resource and an Annotation, even if the client cannot process the representation's format.

For HEAD and GET requests, the response must have an `ETag` header with an entity reference value that implements the notion of entity tags from HTTP [rfc7232]. This value will be used by the client when sending update or delete requests.

The response must have an `Allow` header that lists the HTTP methods available for interacting with the Annotation [rfc7231].

For HEAD and GET requests, if the server supports content negotiation by format or JSON-LD profile, the response must have a `Vary` header with `Accept` in the value [rfc7231]. This is to ensure that caches understand that the representation changes based on the value of that request header.

Request:

EXAMPLE 1

```
GET /annotations/anno1 HTTP/1.1
Host: example.org
Accept: application/ld+json; profile="http://www.w3.org/ns/anno.jsonld
```

Response:

EXAMPLE 2

```
HTTP/1.1 200 OK
Content-Type: application/ld+json; profile="http://www.w3.org/ns/anno.
Link: <http://www.w3.org/ns/ldp#Resource>; rel="type"
ETag: "_87e52ce126126"
Allow: PUT,GET,OPTIONS,HEAD,DELETE,PATCH
Vary: Accept
Content-Length: 287

{
  "@context": "http://www.w3.org/ns/anno.jsonld",
  "id": "http://example.org/annotations/anno1",
  "type": "Annotation",
  "created": "2015-01-31T12:03:45Z",
  "body": {
    "type": "TextualBody",
    "value": "I like this page!"
  },
  "target": "http://www.example.com/index.html"
}
```

4. Annotation Containers

If the Annotation Server supports the management of Annotations, including one or more of creating, updating, and deleting them, then the following section's requirements apply. The

Annotation Protocol is a use of the Linked Data Platform [ldp] specification, with some additional constraints derived from the Web Annotation Data Model [annotation-model].

An Annotation Server must provide one or more Containers within which Annotations can be managed: an Annotation Container. An Annotation Container is at the same time both a Container [ldp] (a service for managing Annotations) and an OrderedCollection [activitystreams-core] (an ordered list of Annotations). It can have descriptive and technical information associated with it to allow clients to present it to a user in order to allow her to decide if it should be used or not. The classes, properties and representations for the Collection model are described in the Web Annotation Data Model, and the mappings to Activity Streams provided in the Web Annotation Vocabulary [annotation-vocab].

Annotation Containers should implement the LDP Basic Container specification, but may instead implement another type of Container, such as a Direct or Indirect Container, to fulfill business needs. The URI of an Annotation Container must not have a query or fragment component, and the path component must end in a `"/"` character.

Implementations should use HTTPS rather than HTTP for all interactions with Annotation Containers.

The creation, management, and structure of Annotation Containers are beyond the scope of this specification. Please see the Linked Data Platform specification [ldp] for additional information.

4.1 Container Retrieval

The Annotation Server must support the following HTTP methods on the Annotation Container's IRI:

- `GET` (retrieve the description of the Container and the list of its contents, described below),
- `HEAD` (retrieve the headers of the Container without an entity-body),
- `OPTIONS` (enable CORS pre-flight requests [cors]).

When an HTTP GET request is issued against the Annotation Container, the server must return a description of the container. That description must be available in JSON-LD, should be available in Turtle, and may be available in other formats. The JSON-LD serialization of the Container's description should use both the LDP context (`http://www.w3c.org/ns/ldp.jsonld`), and the Web Annotation's profile and context [annotation-model], unless the request would determine otherwise.

Servers should use the 200 HTTP status code if the request is successfully completed without errors and does not require redirection based on the client's preferences.

All supported methods for interacting with the Annotation Container should be advertised in the

`Allow` header of the `GET`, `HEAD` and `OPTIONS` responses from the container's IRI . The `Allow` header may also be included on any other responses.

Annotation Containers must return a `Link` header [rfc5988] on all responses with the following components:

- It must advertise its type by including a link where the `rel` parameter value is `type` and the target IRI is the appropriate Container Type, such as `http://www.w3.org/ns/ldp#BasicContainer` for Basic Containers.
- It must advertise that it imposes Annotation protocol specific constraints by including a link where the target IRI is `http://www.w3.org/TR/annotation-protocol/`, and the `rel` parameter value is the IRI `http://www.w3.org/ns/ldp#constrainedBy`.

For HEAD and GET requests, responses from Annotation Containers must include an `ETag` header that implements the notion of entity tags from HTTP [rfc7232]. This value should be used by administrative clients when updating the container by including it in an `If-Match` request header in the same way as clients wanting to update an Annotation.

If the `Accept` header is absent from a GET request, then Annotation Servers must respond with a JSON-LD representation of the Annotation Container, however clients with a preference for JSON-LD should explicitly request it using an `Accept` request header.

If the server supports content negotiation by format or JSON-LD profile, the response to a HEAD or GET request from the Annotation Container must have a `Vary` header that includes `Accept` in the value to ensure that caches can determine that the representation will change based on the value of this header in requests.

Responses from Annotation Containers that support the use of the POST method to create Annotations should include an `Accept-Post` header on responses to GET, HEAD and OPTIONS requests. The value is a comma separated list of media-types that are acceptable for the client to send via POST [ldp].

EXAMPLE 3: Example Annotation Container Headers

```
Link: <http://www.w3.org/ns/ldp#BasicContainer>; rel="type"
Link: <http://www.w3.org/TR/annotation-protocol/>; rel="http://www.w3.
ETag: "0f6b5cd8dc1f754a1738a53b1da34f6b"
Vary: Accept
Allow: POST, GET, OPTIONS, HEAD
Accept-Post: application/ld+json; profile="http://www.w3.org/ns/anno.j
```

4.2 Container Representations

As there are likely to be many Annotations in a single Container, the Annotation Protocol adopts the ActivityStreams collection paging mechanism for returning the contents of the Container. Each Collection Page contains an ordered list with a subset of the managed Annotations, such that if every page is traversed, a client can reconstruct the complete, ordered contents of the container/collection. The number of IRIs or Annotation descriptions included on each page is at the server's discretion, and may be inconsistent between pages. The feature or features by which the Annotations are sorted are not explicit in the response.

The requirements for JSON-LD representation of Annotation Collections are defined in the Web Annotation Data Model, and are summarized here.

The Collection must have an IRI that identifies it, and must have at least the `AnnotationCollection` class (the name associated with OrderedCollection in the JSON-LD context) but may have other types as well, including the type of LDP Container used. It should have a human readable `label`, and may have other properties such as `creator` and `created`.

If there are greater than zero Annotations in the Container, the representation must either include a link to the first page of Annotations as the value of the `first` property, or include the representation of the first page embedded within the response. If there is more than one page of Annotations, then the representation should have a link to the last page using the `last` property.

The representation of the Container should include the `total` property with the total number of annotations in the Container. The Container should include the `modified` property with the most recent timestamp of when any of the annotations in the Container. This timestamp allows clients to detect when to re-cache data, even if there are the same number of annotations as the same number may have been added and deleted.

The IRI of the Container provided in the response should differentiate between whether the pages contain just the IRIs, or the full descriptions of the Annotations. It is recommended that this be done with a query parameter. The server may redirect the client to this IRI and deliver the response there, otherwise it must include a `Content-Location` header with the IRI as its value.

4.2.1 Container Representation Preferences

There are three preferences for Container requests that will govern the representation of the server's responses:

1. If the client prefers to only receive the Container description and no Annotations (either URI or full descriptions) embedded in the Container response, then it must include a `Prefer` request header with the value `return=representation; include="http://www.w3.org/ns/ldp#PreferMinimalContainer"`.
2. If the client prefers to receive the Annotations only as IRI references, either embedded in the

current Container response or future paged responses, then it must include a `Prefer` request header with the value `return=representation;include="http://www.w3.org/ns/oa#PreferContainedIRIs"`.

3. If the client prefers to receive complete Annotation descriptions, either in the current Container response or future paged responses, then it must include a `Prefer` request header with the value `return=representation;include="http://www.w3.org/ns/oa#PreferContainedDescriptions"`.

The client may send multiple preferences as the value of the `include` parameter as defined by the Linked Data Platform [ldp]. However, the client must not include both the `PreferContainedIRIs` and `PreferContainedDescriptions` preferences on the same request, as the server cannot honor both at the same time. If the `PreferMinimalContainer` preference is given, then the server should not embed the Annotations or references to them, but should include a reference to the first and last Annotation Pages. Whether the pages are of IRI references or complete descriptions is governed by the use of `PreferContainedIRIs` and `PreferContainedDescriptions` respectively. If no preference is given by the client, the server should default to the `PreferContainedDescriptions` behavior. The server may ignore the client's preferences.

4.2.2 Representations without Annotations

If the client requests the minimal representation of an Annotation Container, the response must not include either the `ldp:contains` predicate nor embed the first page of Annotations within the response.

The linked pages should follow any `PreferContainedDescriptions` or `PreferContainedIRIs` preferences.

The server may return a representation without embedded Annotations, even if the `PreferMinimalContainer` preference is not supplied.

Request:

EXAMPLE 4: Container Request without Annotations

```
GET /annotations/ HTTP/1.1
Host: example.org
Accept: application/ld+json; profile="http://www.w3.org/ns/anno.jsonld
Prefer: return=representation;include="http://www.w3.org/ns/ldp#Prefer
```

Response:

EXAMPLE 5: Container Response without Annotations

```
HTTP/1.1 200 OK
Content-Type: application/ld+json; profile="http://www.w3.org/ns/anno.
ETag: "_87e52ce123123"
Link: <http://www.w3.org/ns/ldp#BasicContainer>; rel="type"
Link: <http://www.w3.org/TR/annotation-protocol/>; rel="http://www.w3.
Allow: POST,GET,OPTIONS,HEAD
Vary: Accept
Content-Length: 368

{
  "@context": [
    "http://www.w3.org/ns/anno.jsonld",
    "http://www.w3.org/ns/ldp.jsonld"
  ],
  "id": "http://example.org/annotations/?iris=1",
  "type": ["BasicContainer", "AnnotationCollection"],
  "total": 42023,
  "modified": "2016-07-20T12:00:00Z",
  "label": "A Container for Web Annotations",
  "first": "http://example.org/annotations/?iris=1&page=0",
  "last": "http://example.org/annotations/?iris=1&page=42"
}
```

4.2.3 Representations with Annotation IRIs

If the Server supports Container preferences, it must respond to `PreferContainedIRIs` with a response containing an `AnnotationPage` as the value of `first` with its `items` containing only the IRIs of the contained Annotations.

The linked pages should follow the `PreferContainedIRIs` preference.

The `PreferContainedIRIs` and the `PreferContainedDescriptions` preferences are mutually exclusive.

Request for embedded IRIs:

EXAMPLE 6: Container Request (Embedded IRIs)

```
GET /annotations/ HTTP/1.1
Host: example.org
Accept: application/ld+json; profile="http://www.w3.org/ns/anno.jsonld
Prefer: return=representation;include="http://www.w3.org/ns/oa#PreferC
```

Response:

EXAMPLE 7: Container Response (Embedded IRIs)

```
HTTP/1.1 200 OK
Content-Type: application/ld+json; profile="http://www.w3.org/ns/anno.
Content-Location: http://example.org/annotations/?iris=1
ETag: "_87e52ce123123"
Link: <http://www.w3.org/ns/ldp#BasicContainer>; rel="type"
Link: <http://www.w3.org/TR/annotation-protocol/>; rel="http://www.w3.
Allow: POST,GET,OPTIONS,HEAD
Vary: Accept, Prefer
Content-Length: 397

{
  "@context": [
    "http://www.w3.org/ns/anno.jsonld",
    "http://www.w3.org/ns/ldp.jsonld"
  ],
  "id": "http://example.org/annotations/?iris=1",
  "type": ["BasicContainer", "AnnotationCollection"],
  "total": 42023,
  "modified": "2016-07-20T12:00:00Z",
  "label": "A Container for Web Annotations",
  "first": {
    "id": "http://example.org/annotations/?iris=1&page=0",
    "type": "AnnotationPage",
    "next": "http://example.org/annotations/?iris=1&page=1",
    "items": [
      "http://example.org/annotations/anno1",
      "http://example.org/annotations/anno2",
      "http://example.org/annotations/anno3",
      "http://example.org/annotations/anno4",
      "http://example.org/annotations/anno5",
      "http://example.org/annotations/anno6",
      ...
      "http://example.org/annotations/anno999",
    ]
  },
  "last": "http://example.org/annotations/?iris=1&page=42"
}
```

4.2.4 Representations with Annotation Descriptions

If the Server supports Container preferences, it must respond to

`PreferContainedDescriptions` with a response containing an `AnnotationPage` as the value of `first` with its `items` containing complete, inline Annotations.

The linked pages should follow the `PreferContainedDescriptions` preference.

The `PreferContainedIRIs` and the `PreferContainedDescriptions` preferences are mutually exclusive.

Request for embedded descriptions:

> EXAMPLE 8: Container Request (Embedded Descriptions)
>
> ```
> GET /annotations/ HTTP/1.1
> Host: example.org
> Accept: application/ld+json; profile="http://www.w3.org/ns/anno.jsonld
> Prefer: return=representation;include="http://www.w3.org/ns/oa#PreferC
> ```

EXAMPLE 9: Container Response (Embedded Descriptions)

```
HTTP/1.1 200 OK
Content-Type: application/ld+json; profile="http://www.w3.org/ns/anno.
Allow: GET,OPTIONS,HEAD
Vary: Accept, Prefer
Content-Length: 924

{
  "@context": [
    "http://www.w3.org/ns/anno.jsonld",
    "http://www.w3.org/ns/ldp.jsonld"
  ],
  "id": "http://example.org/annotations/?iris=0",
  "type": ["BasicContainer", "AnnotationCollection"],
  "total": 42023,
  "modified": "2016-07-20T12:00:00Z",
  "label": "A Container for Web Annotations",
  "first": {
    "id": "http://example.org/annotations/?iris=0&page=0",
    "type": "AnnotationPage",
    "next": "http://example.org/annotations/?iris=0&page=1",
    "items": [
      {
        "id": "http://example.org/annotations/anno1",
        "type": "Annotation",
        "body": "http://example.net/body1",
        "target": "http://example.com/page1"
      },
      {
        "id": "http://example.org/annotations/anno2",
        "type": "Annotation",
        "body": {
          "type": "TextualBody",
          "value": "I like this!"
        },
        "target": "http://example.com/book1"
      },
      ...
      {
        "id": "http://example.org/annotations/anno50",
        "type": "Annotation",
        "body" : "http://example.org/texts/description1",
        "target": "http://example.com/images/image1"
      }
    ]
  },
  "last": "http://example.org/annotations/?iris=0&page=840"
```

```
}
```

4.3 Annotation Pages

Individual pages are instances of the Activity Streams `OrderedCollectionPage` class, which is refered to as `AnnotationPage` in the Web Annotation JSON-LD context. The page contains the Annotations, either via their IRIs or full descriptions, in the `items` property.

The requirements for JSON-LD representation of Annotation Collections Pages are defined in the Web Annotation Data Model, and are summarized here.

The Annotation Page must have an IRI that identifies it, and must have at least the `AnnotationPage` class but may have other types as well. If the Page is not the last Page in the Collection, then it must have a reference to the Page which follows it using the `next` property. If the Page is not the first Page in the Collection, it should have a reference of the previous Page using the `prev` property. Pages should give the position of the first Annotation in the `items` list relative to the order of the Collection using the zero-based `startIndex` property.

Each page must have a link to the Collection that it is part of, using the `partOf` property. The description of the Collection should include both the `total` and `modified` properties. The response may include other properties of the Collection in the response, such as the `label` or `first` and `last` links.

The client should not send the `Prefer` header when requesting the Page, as it has already been taken into account when requesting the Collection.

This specification does not require any particular functionality when a client makes requests other than GET, HEAD or OPTIONS to a page.

As the Page is not an LDP Container, it does not have the requirement to include a `Link` header with a type. That the URLs could be constructed with query parameters added to the Container's IRI is an implementation convenience, and does not imply the type of the resource.

Embedded IRIs Interaction Example

Request:

EXAMPLE 10: Page Request

```
GET /annotations/?iris=1&page=0 HTTP/1.1
Host: example.org
Accept: application/ld+json; profile="http://www.w3.org/ns/anno.jsonld
```

Response:

EXAMPLE 11: Page Response (Embedded IRIs)

```
HTTP/1.1 200 OK
Content-Type: application/ld+json; profile="http://www.w3.org/ns/anno.
Allow: GET,OPTIONS,HEAD
Vary: Accept
Content-Length: 630

{
  "@context": "http://www.w3.org/ns/anno.jsonld",
  "id": "http://example.org/annotations/?iris=1&page=0",
  "type": "AnnotationPage",
  "partOf": {
    "id": "http://example.org/annotations/?iris=1",
    "total": 42023,
    "modified": "2016-07-20T12:00:00Z",
  },
  "startIndex": 0,
  "next": "http://example.org/annotations/?iris=1&page=1",
  "items": [
    "http://example.org/annotations/anno1",
    "http://example.org/annotations/anno2",
    "http://example.org/annotations/anno3",
    "http://example.org/annotations/anno4",
    "http://example.org/annotations/anno5",
    "http://example.org/annotations/anno6",
    ...
    "http://example.org/annotations/anno999",
  ]
}
```

Embedded Descriptions Interaction Example

Request:

EXAMPLE 12: Page Request (Embedded Descriptions)

```
GET /annotations/?iris=0&page=0 HTTP/1.1
Host: example.org
Accept: application/ld+json; profile="http://www.w3.org/ns/anno.jsonld
```

EXAMPLE 13: Page Response (Embedded Descriptions)

```
HTTP/1.1 200 OK
Content-Type: application/ld+json; profile="http://www.w3.org/ns/anno.
Allow: GET,OPTIONS,HEAD
Vary: Accept, Prefer
Content-Length: 924

{
  "@context": "http://www.w3.org/ns/anno.jsonld",
  "id": "http://example.org/annotations/?iris=0&page=0",
  "type": "AnnotationPage",
  "partOf": {
    "id": "http://example.org/annotations/?iris=0",
    "total": 42023,
    "modified": "2016-07-20T12:00:00Z",
  },
  "startIndex": 0,
  "next": "http://example.org/annotations/?iris=0&page=1",
  "items": [
    {
      "id": "http://example.org/annotations/anno1",
      "type": "Annotation",
      "body": "http://example.net/body1",
      "target": "http://example.com/page1"
    },
    {
      "id": "http://example.org/annotations/anno2",
      "type": "Annotation",
      "body": {
        "type": "TextualBody",
        "value": "I like this!"
      },
      "target": "http://example.com/book1"
    },
    ...
    {
      "id": "http://example.org/annotations/anno50",
      "type": "Annotation",
      "body" : "http://example.org/texts/description1",
      "target": "http://example.com/images/image1"
    }
  ]
}
```

4.4 Discovery of Annotation Containers

As the IRI for Annotation Containers may be any IRI, and it is unlikely that every Web Server will support the functionality, it is important to be able to discover the availability of these services.

Any resource may link to an Annotation Container when Annotations on the resource should be created within the referenced Container. This link is carried in an HTTP `Link` header and the value of the `rel` parameter must be `http://www.w3.org/ns/oa#annotationService`.

For HTML representations of resources, the equivalent `link` tag in the header of the document may also be used.

For an example image resource, a GET request and response with a link to the above Annotation Container might look like:

Request:

EXAMPLE 14

```
GET /images/logo.jpg HTTP/1.1
Host: example.com
```

Response:

EXAMPLE 15

```
HTTP/1.1 200 OK
Content-Type: image/jpeg
Link: <http://example.org/annotations/>; rel="http://www.w3.org/ns/oa#
Allow: GET
Content-Length: 76983

[...]
```

5. Creation, Updating and Deletion of Annotations

5.1 Create a New Annotation

New Annotations are created via a POST request to an Annotation Container. The Annotation, serialized as JSON-LD, is sent in the body of the request. All of the known information about the

Annotation should be sent, and if there are already IRIs associated with the resources, they should be included. The serialization should use the Web Annotation JSON-LD profile, and servers may reject other contexts even if they would otherwise produce the same model. The server may reject content that is not considered an Annotation according to the Web Annotation specification [annotation-model].

Upon receipt of an Annotation, the server may assign IRIs to any resource or blank node in the Annotation, and must assign an IRI to the Annotation resource in the `id` property, even if it already has one provided. The server should use HTTPS IRIs when those resources are able to be retrieved individually. The IRI for the Annotation must be the IRI of the Container with an additional component added to the end.

The server may add information to the Annotation. Possible additional information includes the agent that created it, the time of the Annotation's creation, or additional types and formats of the constituent resources.

If the Annotation contains a `canonical` property, then that reference must be maintained without change. If the Annotation has an IRI in the `id` property, then it should be copied to the `via` property, and the IRI assigned by the server at which the Annotation will be available must be put in the `id` field to replace it.

The server must respond with a `201` Created response if the creation is successful, and an appropriate error code otherwise. The response must have a `Location` header with the Annotation's new IRI.

Request:

EXAMPLE 16

```
POST /annotations/ HTTP/1.1
Host: example.org
Accept: application/ld+json; profile="http://www.w3.org/ns/anno.jsonld
Content-Type: application/ld+json; profile="http://www.w3.org/ns/anno.
Content-Length: 202

{
  "@context": "http://www.w3.org/ns/anno.jsonld",
  "type": "Annotation",
  "body": {
    "type": "TextualBody",
    "value": "I like this page!"
  },
  "target": "http://www.example.com/index.html"
}
```

Response:

EXAMPLE 17

```
HTTP/1.1 201 CREATED
Allow: PUT,GET,OPTIONS,HEAD,DELETE,PATCH
Location: http://example.org/annotations/anno1
Content-Type: application/ld+json; profile="http://www.w3.org/ns/anno.
Content-Length: 287
ETag: "_87e52ce126126"

{
  "@context": "http://www.w3.org/ns/anno.jsonld",
  "id": "http://example.org/annotations/anno1",
  "type": "Annotation",
  "created": "2015-01-31T12:03:45Z",
  "body": {
    "type": "TextualBody",
    "value": "I like this page!"
  },
  "target": "http://www.example.com/index.html"
}
```

5.2 Suggesting an IRI for an Annotation

The IRI path segment that is appended to the Container IRI for a resource may be suggested by the Annotation Client by using the `Slug` HTTP header on the request when the resource is created. The server should use this name, so long as it does not already identify an existing resource, but may ignore it and use an automatically assigned name.

Request:

EXAMPLE 18

```
POST /annotations/ HTTP/1.1
Host: example.org
Accept: application/ld+json; profile="http://www.w3.org/ns/anno.jsonld
Content-Type: application/ld+json; profile="http://www.w3.org/ns/anno.
Content-Length: 202
Slug: "my_first_annotation"

{
  "@context": "http://www.w3.org/ns/anno.jsonld",
  "type": "Annotation",
  "body": {
    "type": "TextualBody",
    "value": "I like this page!"
  },
  "target": "http://www.example.com/index.html"
}
```

Response:

EXAMPLE 19

```
HTTP/1.1 201 CREATED
Link: <http://www.w3.org/ns/ldp#BasicContainer>; rel="type"
Allow: PUT,GET,OPTIONS,HEAD,DELETE,PATCH
Location: http://example.org/annotations/my_first_annotation
Content-Type: application/ld+json; profile="http://www.w3.org/ns/anno.
ETag: "_87e52ce126126"
Vary: Accept
Content-Length: 301

{
  "@context": "http://www.w3.org/ns/anno.jsonld",
  "id": "http://example.org/annotations/my_first_annotation",
  "type": "Annotation",
  "created": "2015-01-31T12:03:45Z",
  "body": {
    "type": "TextualBody",
    "value": "I like this page!"
  },
  "target": "http://www.example.com/index.html"
}
```

5.3 Update an Existing Annotation

Annotations can be updated by using a PUT request to replace the entire state of the Annotation. Annotation Servers should support this method. Servers may also support using a PATCH request to update only the aspects of the Annotation that have changed, but that functionality is not specified in this document.

Replacing the Annotation with a new state must be done with the PUT method, where the body of the request is the intended new state of the Annotation. The client should use the `If-Match` header with a value of the ETag it received from the server before the editing process began, to avoid collisions of multiple users modifying the same Annotation at the same time. This feature is not mandatory to support, as not every system will have multiple users with the potential to change a single Annotation, or use cases might dictate situations in which overwriting is the desired behavior.

Servers should reject update requests that modify the values of the `canonical` or `via` properties, if they have been already set, unless business logic allows the request to be trusted as authoritatively correctly a previous error.

If successful, the server must return a 200 OK status with the Annotation as the body according to the content-type requested. As with creation, the server must return the new state of the Annotation in the response.

Request:

EXAMPLE 20

```
PUT /annotations/anno1 HTTP/1.1
Host: example.org
Accept: application/ld+json; profile="http://www.w3.org/ns/anno.jsonld
Content-Type: application/ld+json; profile="http://www.w3.org/ns/anno.
Content-Length: 294
If-Match: "_87e52ce126126"

{
  "@context": "http://www.w3.org/ns/anno.jsonld",
  "id": "http://example.org/annotations/anno1",
  "type": "Annotation",
  "created": "2015-02-01T10:13:40Z",
  "body": {
    "type": "TextualBody",
    "value": "I REALLY like this page!"
  },
  "target": "http://www.example.com/index.html"
}
```

Response:

EXAMPLE 21

```
HTTP/1.1 200 OK
Content-Type: application/ld+json; profile="http://www.w3.org/ns/anno.
ETag: "_87e52ce234234"
Link: <http://www.w3.org/ns/ldp#Resource>; rel="type"
Allow: PUT,GET,OPTIONS,HEAD,DELETE,PATCH
Vary: Accept
Content-Length: 331

{
  "@context": "http://www.w3.org/ns/anno.jsonld",
  "id": "http://example.org/annotations/anno1",
  "type": "Annotation",
  "created": "2015-02-01T10:13:40Z",
  "modified": "2015-02-02T20:43:19Z",
  "body": {
    "type": "TextualBody",
    "value": "I REALLY like this page!"
  },
  "target": "http://www.example.com/index.html"
}
```

5.4 Delete an Existing Annotation

Clients must use the DELETE HTTP method to request that an Annotation be deleted by the server. Annotation Servers should support this method. Clients should send the ETag of the Annotation in the `If-Match` header to ensure that it is operating against the most recent version of the Annotation.

If the DELETE request is successfully processed, then the server must return a 204 status response. The IRIs of deleted Annotations should not be re-used for subsequent Annotations. The IRI of the deleted Annotation must be removed from the Annotation Container it was created in. There are no requirements made on the body of the response, and it may be empty.

Request:

EXAMPLE 22

```
DELETE /annotations/anno1 HTTP/1.1
Host: example.org
If-Match: "_87e52ce126126"
```

Response:

EXAMPLE 23

```
HTTP/1.1 204 NO CONTENT
Content-Length: 0
```

6. Error Conditions

This section is non-normative.

There are inevitably situations where errors occur when retrieving or managing Annotations. The use of the HTTP status codes below provides a method for clients to understand the reason why a request has failed. Some of the situations that might occur, and the preferred HTTP status code are given below. This list is intended to be informative and explanatory, rather than imposing additional requirements beyond those already established by HTTP.

Code	Example Situation
400	The Annotation Client sent a request which the Annotation Server cannot process due to the request not following the appropriate specifications.
401	The Annotation Client is not authorized to perform the requested operation, such as creating or deleting an Annotation, as it did not supply authentication credentials.
403	The Annotation Client is not authorized to perform the requested operation, as the authentication credentials supplied did not meet the requirements of a particular access control policy for the Annotation or Annotation Container.
404	The Annotation or Annotation Container requested does not exist.
405	The requested HTTP method is not allowed for the resource, such as trying to POST to an Annotation Container page, or trying to PATCH an Annotation when that functionality is not supported.

406	The requested format for the Annotation or Annotation Container's representation is not available, for example if a client requested RDF/XML and the server does not support that (optional) transformation.
409	The Annotation Client tried to set or change a value that the server does not allow Clients to modify, such as the containment list of an Annotation Container or server set modification timestamps.
410	The Annotation is known to have existed in the past and was deleted.
412	The Annotation Client supplied an If-Match header that did not match the ETag of the Annotation being modified.
415	The Annotation Client sent an entity-body that is not able to be processed by the Server, such as non-Annotation or in a context that is unrecognized.

7. Containers for Related Resources

Annotations may have related resources that are required for their correct interpretation and rendering, such as content resources used in or as the Body, CSS stylesheets that determine the rendering of the annotation, SVG documents describing a non-rectangular region of a resource, and so forth. If these resources do not already have IRIs, then they need to be made available somewhere so that they can be referred to.

Annotation Servers may support the management of related resources independently from the Annotations. If a server supports the management of these resources, it should do this with one or more separate Containers. Resources that are not Annotations should not be included in an Annotation Container, as Annotation Clients would not expect to find arbitrary content when dereferencing the IRIs. Containers for related resources may contain both RDF Sources and Non-RDF Sources. No restrictions are placed on the type or configuration of the Container beyond those of the Linked Data Platform [ldp].

Containers for related resources must support the same HTTP methods as described above for the Annotation Container, and must support identifying their type with a `Link` header. The `constrainedBy` link header on the response when dereferencing the Container should refer to a server specific set of constraints listing the types of content that are acceptable.

A. Candidate Recommendation Exit Criteria

This section is non-normative.

For this specification to be advanced to Proposed Recommendation, there must be at least two

independent implementations of each feature described below. Each feature may be implemented by a different set of products, and there is no requirement that any single product implement every feature.

Features

For the purposes of evaluating exit criteria, the following operations are considered as features:

- HTTP GET of an Annotation
- HTTP GET of an Annotation Collection
- HTTP GET of an Annotation Collection Page, with embedded Annotations
- HTTP GET of an Annotation Collection Page, without embedded Annotations
- POST of an Annotation to an Annotation Collection
- POST of an Annotation to an Annotation Collection, with a Slug to suggest the IRI
- PUT of an Annotation to update an existing Annotation in an Annotation Collection
- DELETE of an Annotation

Each feature must be implemented according to the requirements given in the specification, regarding the HTTP headers, status codes, and entity body. Software that does not alter its behavior in the presence or lack of a given feature is not deemed to implement that feature for the purposes of exiting the Candidate recommendation phase.

B. Changes from Previous Versions

This section is non-normative.

B.1 Changes from the Proposed Recommendation of 2017-01-17

No significant changes.

B.2 Changes from the Candidate Recommendation of 2016-09-06

Editorial changes in this specification from the Candidate Recommendation of 2016-09-06 are:

- Clarified which header requirements are appropriate for HEAD/GET, rather than OPTIONS.
- Clarified interaction of multiple preferences in a single request.
- Removed incorrect Prefer from the Vary header from the pages example.

- Summarized requirements from the Web Annotation Data Model for Collections and Pages
- Clarified text regarding URI requirements for Containers.
- Clarified use of modified with respect to Containers and Pages.
- Clarified use of canonical with update operations.
- Fixed RFC 7230 / RFC 7232 typo.
- Clarified use of ETag and If-Match.

B.3 Changes from the Candidate Recommendation of 2016-07-12

Editorial changes in this specification from the Candidate Recommendation of 2016-07-12 are:

- Added CR Exit Criteria
- Editorial restructuring of the Pagination content.
- Clarified requirements for Annotation Server's representation preference handling.
- Explained the proper use of the `Prefer` header when the client has multiple preferences.

B.4 Changes from the Working Draft of 2016-03-31

Significant technical changes in this specification from the Working Draft Published of 2016-03-31 are:

- Use ActivityStreams based Paging mechanism to replace LDP Paging, to allow for in-page order.
- Recommend the use of HTTPS over HTTP.
- Rename PreferContainedURIs to PreferContainedIRIs.
- Add recommendation for Accept-Post, with example.
- Clarify expected status codes for successful interactions.
- Restructure Container Retrieval section and promote Container Representations and Annotation Pages sections.

C. Acknowledgments

The Web Annotation Working Group gratefully acknowledges the contributions of the Open Annotation Community Group. The output of the Community Group was fundamental to the current data model and protocol.

The following people have been instrumental in providing thoughts, feedback, reviews, content, criticism and input in the creation of this specification:

Vladimir Alexiev, Art Barstow, Tim Berners-Lee, Chris Birk, Dan Brickley, Sarven Capadisli, Paolo Ciccarese, Tim Cole, Ray Denenberg, TB Dinesh, Sergiu Gordea, Benjamin Goering, Amy Guy, Ivan Herman, Frederick Hirsch, Antoine Isaac, Jacob Jett, Takeshi Kanai, Gregg Kellogg, Andreas Kuckartz, Randall Leeds, Hugo Manguinhas, Shane McCarron, Ben De Meester, Luc Moreau, Mark Nottingham, Addison Phillips, Davis Salisbury, Robert Sanderson, Felix Sasaki, Doug Schepers, Tzviya Siegman, Stian Soiland-Reyes, Manu Sporny, Nick Stenning, Jon Stroop, Lutz Suhrbier, Kyrce Swenson, Raphaël Troncy, Simeon Warner, Erik Wilde, Dan Whaley, Benjamin Young

D. References

D.1 Normative references

[RFC2119]
Key words for use in RFCs to Indicate Requirement Levels. S. Bradner. IETF. March 1997. Best Current Practice. URL: https://tools.ietf.org/html/rfc2119

[activitystreams-core]
Activity Streams 2.0. James Snell; Evan Prodromou. W3C. 15 December 2016. W3C Candidate Recommendation. URL: https://www.w3.org/TR/activitystreams-core/

[activitystreams-vocabulary]
Activity Vocabulary. James Snell; Evan Prodromou. W3C. 15 December 2016. W3C Candidate Recommendation. URL: https://www.w3.org/TR/activitystreams-vocabulary/

[annotation-model]
Web Annotation Data Model. Robert Sanderson; Paolo Ciccarese; Benjamin Young. W3C. W3C Recommendation. URL: https://www.w3.org/TR/annotation-model/

[annotation-vocab]
Web Annotation Vocabulary. Robert Sanderson; Paolo Ciccarese; Benjamin Young. W3C. W3C Recommendation. URL: https://www.w3.org/TR/annotation-vocab/

[cors]
Cross-Origin Resource Sharing. Anne van Kesteren. W3C. 16 January 2014. W3C Recommendation. URL: https://www.w3.org/TR/cors/

[ldp]
Linked Data Platform 1.0. Steve Speicher; John Arwe; Ashok Malhotra. W3C. 26 February 2015. W3C Recommendation. URL: https://www.w3.org/TR/ldp/

[rfc3987]
Internationalized Resource Identifiers (IRIs). M. Duerst; M. Suignard. IETF. January 2005.

Proposed Standard. URL: https://tools.ietf.org/html/rfc3987

[rfc5988]
Web Linking. M. Nottingham. IETF. October 2010. Proposed Standard. URL: https://tools.ietf.org/html/rfc5988

[rfc7231]
Hypertext Transfer Protocol (HTTP/1.1): Semantics and Content. R. Fielding, Ed.; J. Reschke, Ed.. IETF. June 2014. Proposed Standard. URL: https://tools.ietf.org/html/rfc7231

[rfc7232]
Hypertext Transfer Protocol (HTTP/1.1): Conditional Requests. R. Fielding, Ed.; J. Reschke, Ed.. IETF. June 2014. Proposed Standard. URL: https://tools.ietf.org/html/rfc7232

↥