webmention-spec

From IndieWeb

Status
This is an **Editor's Draft** yet mature enough to encourage implementations and feedback.
Latest Published Version
http://webmention.net/
Latest Editor's Draft
http://webmention.net/draft/
Participate
Issues (https://github.com/aaronpk/webmention/issues)
IRC: discussion on #indieweb-dev (https://chat.indieweb.org/dev) #indieweb-dev on Libera (irc://irc.libera.chat/indieweb-dev)
Editors
Aaron Parecki (http://aaronparecki.com)
Authors
Other contributors: [1] (https://indiewebcamp.com/wiki/index.php?title=Webmention&action=history) [2] (https://indiewebcamp.com/wiki/index.php?title=webmention-spec&action=history)
License

Abstract

Webmention is a simple way to notify any URL when you link to it on your site. From the receiver's perspective, it's a way to request notifications when other sites link to it.

Contents

Webmention Protocol

Sending Webmentions

Sender discovers receiver webmention endpoint

The sender MUST fetch the target URL and check for an HTTP Link header with `rel="webmention"`, or a `<link>` or `<a>` element with `rel="webmention"`. If more than one of these is present, the first HTTP Link header takes precedence, followed by the first `<link>` element, and finally the first `<a>` element. Clients MUST support all three options and fall back in this order. Senders MAY initially

make an HTTP HEAD request to check for the Link header before making a GET request.

```
GET /post-by-aaron HTTP/1.1
Host: aaronpk.com
```

```
HTTP/1.1 200 OK
Link: <http://aaronpk.com/webmention-endpoint>; rel="webmention"

<html>
<head>
...
<link href="http://aaronpk.com/webmention-endpoint" rel="webmention" />
...
</head>
<body>
....
<a href="http://aaronpk.com/webmention-endpoint" rel="webmention" />
...
</body>
</html>
```

Sender notifies receiver

The sender MUST post x-www-form-urlencoded `source` and `target` parameters to the webmention endpoint, where `source` is the URL of the sender's page containing a link, and the `target` is the URL of the page being linked to.

The webmention endpoint will validate and process the request, and return an HTTP status code. Most often, `202 Accepted` or `201 Created` will be returned, indicating that the request is queued and being processed asynchronously to prevent DoS attacks. If the response code is 201, the Location header will include a URL that can be used to monitor the status of the request.

Any `2xx` response code MUST be considered a success.

```
POST /webmention-endpoint HTTP/1.1
Host: aaronpk.com
Content-Type: application/x-www-form-urlencoded

source=https://waterpigs.co.uk/post-by-barnaby&
target=https://aaronpk.com/post-by-aaron
```

```
HTTP/1.1 202 Accepted
```

Receiving Webmentions

Upon receipt of a POST request containing the `source` and `target` parameters, the receiver SHOULD queue and process the request asynchronously to prevent DoS attacks. There are three possible responses to the request, depending on how the receiver processes it.

If the receiver creates a status page which the sender can use to check the status, the receiver MUST reply with an HTTP `201 Created` response with a `Location` header pointing to the status URL. The response body MAY contain content, in which case a human readable response is recommended.

```
HTTP/1.1 201 Created
Location: http://aaronparecki.com/webmention/DEhB9Jme
Content-type: text/plain
```

```
The webmention is being processed. You can check on its status here: http://aaronparecki.com/webmention/DEhB9Jme
```

If the receiver processes the request asynchronously but does not return a status URL, the receiver MUST reply with an HTTP `202 Accepted` response. The response body MAY contain content, in which case a human readable response is recommended.

```
HTTP/1.1 202 Accepted
Content-type: text/plain

The webmention is being processed
```

If the receiver chooses to process the request and perform the verification step synchronously (not recommended), it MUST respond with a 200 OK status on success.

Request Verification

The receiver MUST check that source and target are valid URLs and are of schemes that are supported by the receiver. (Most commonly this means checking that the source and target schemes are http or https).

The receiver SHOULD check that target is a valid resource that it can accept webmentions for. This check SHOULD happen synchronously to reject invalid webmentions before more in-depth verification begins.

Webmention Verification

Webmention verification SHOULD be handled asynchronously to prevent DoS attacks.

If the receiver is going to use the webmention in some way, (displaying it as a comment on a post, incrementing a "like" counter, notifying the author of a post), then it MUST perform a HTTP `GET` request on source, and follow any HTTP redirects (up to a self-imposed limit such as 20) to confirm that it actually links to `target`.

The receiver SHOULD use per-media-type rules to determine whether the source document links to the target URL. For example, in an HTML document, the receiver should look for `<a href="*">`, `<img href="*">`, `<video src="*">` and other similar links. In a JSON document, the receiver should look for properties whose values are an exact match for the URL. If the document is plain text, the receiver should look for the URL by searching for the string. Other content types may be handled at the implementer's discretion. The source document MUST have an exact match of the target URL provided in order for it to be considered a valid link.

At this point, the receiver MAY publish content from this webmention on the target page or other pages, along with any other data it picks up from source. For example, the receiver may display the contents of the source as a comment on the post.

Error Responses

If the webmention was not successful because of something the *sender* did, it MUST return a 400 Bad Request status code and MAY include a description of the error in the response body.

Possible sender-related errors that can be returned synchronously before making a GET request to the source:

- Specified target URL not found.
- Specified `target` URL does not accept webmentions.

- `source` URL was malformed or is not a supported URL scheme (e.g. a mailto: link)

Possible sender-related errors that can occur after fetching the contents of the source URL:

- `source` URL not found.
- Source URL does not contain a link to the `target` URL.

If the webmention was not successful because of an error on the *receiver's server*, it SHOULD return a 500 Internal Server Error status code and MAY include a description of the error in the response body.

Updating existing webmentions

If receiver had received a webmention in the past with the same `source` and `target` then,

- If both the verification steps are successful, it SHOULD update any existing data it picked from `source` for the existing webmention.
    - When a webmention implementation does support updating (AKA a "webmention update implementation"), it MUST support updating data from properties of the primary object of the source (e.g. properties of the h-entry of the page).
        - A webmention update implementation MAY support updating data from the h-entry of the page. If an implementation does support this, it MUST support it according to the microformats2-parsing and h-entry specifications.
        - A webmention update implementation MAY support updating data from children, or other descendant objects of the primary object (e.g. a comment h-entry inside the h-entry of the page). If an implementation does support this, it MUST support it according to the Salmentions extension specification (AKA a "salmentions implementation").
            - A salmentions implementation MAY support updating data from children of the h-entry of the page. If an implementation does support this, it MUST support it according to the microformats2-parsing and h-entry specifications.
- If it received a `410 Gone` status code on step 2 (performing a `GET` request on source), or received a 200 status code and does not find a link to `target` on `source`, it SHOULD delete the existing webmention.
- Processing webmentions SHOULD be idempotent. For example, receiving multiple webmentions for the same source and target with no content changes should not show as multiple replies.

Security Considerations

Preventing Abuse

- The verification process SHOULD be queued and processed asynchronously to prevent DDoS attacks.
- Receivers SHOULD moderate Webmentions before displaying them
- Receivers MAY periodically re-verify webmentions and update them.
- If a receiver chooses to publish data it picks up from `source`, it should ensure that the data is encoded and/or filtered to prevent XSS and CSRF attacks.

Limits on GET requests

The Webmention protocol relies on the sender making a GET (or HEAD) request to discover the receiver's endpoint, followed by the receiver making a GET request to the sender's web page to verify the link. This means a sender can cause a receiver to make GET requests to arbitrary URLs, opening up a potential DoS vector.

Receivers should place limits on the amount of data and time they spend fetching unverified source URLs. For example, if a source URL doesn't respond within 5 seconds, it can treat that as a failure. Similarly, the receiver can fetch only the first 1mb of the page, since any reasonably HTML or JSON page will be smaller than that.

Extensions

The following Webmention Extension Specifications have 2+ interoperable implementations live on the web and are thus listed here:

- Vouch
- Salmention

Acknowledgements

Thanks to Sandeep Shetty (http://sandeep.io/) for contributing the original draft of the webmention specification.

See Also

- Webmention
- FAQ
- Implementation Guide
- Implementation Details
- Brainstorming

- This page was last edited on 26 May 2021, at 08:41.

- Content is available under a CC0 public domain dedication unless otherwise noted.