Advanced Data Protection Control (ADPC)

Unofficial Draft 15 June 2021

Editors:

Gerben (noyb)

Soheil Human (Sustainable Computing Lab)

Authors:

Soheil Human (Sustainable Computing Lab)

Max Schrems (noyb)

Alan Toner (noyb)

Gerben (noyb)

Ben Wagner (Sustainable Computing Lab)

Abstract

This specification defines a mechanism for expressing user decisions about personal data processing under the European Union's data protection regulations, and similar regulations outside the EU. The mechanism functions through the exchange of HTTP headers between the user agent and the web server, or through an equivalent JavaScript interface.

The mechanism serves as an automated means for users to give or refuse consent, to withdraw any consent already given, as well as to object to processing. The mechanism provides an alternative to existing non-automated consent management approaches (e.g. 'cookie banners') and aims to reduce the efforts of the different parties involved regarding the protection of users' privacy.

Table of Contents

1. Introduction §

This section is non-normative.

Legal frameworks such as the European Union’s General Data Protection Regulation (GDPR) and ePrivacy Directive define rights and obligations around the processing of personal data. The starting position of the GDPR is that processing of personal data is only lawful if it has an appropriate legal basis; one basis being that *“the data subject has given consent to the processing of his or her personal data for one or more specific purposes”* (point (a) of Article 6(1) GDPR). Similarly, the ePrivacy Directive (in Article 5(3)) requires the user’s consent when any data is stored on or retrieved from terminal equipment beyond what is strictly necessary. Moreover, when a data controller relies on legitimate interest as the legal basis for direct marketing, the user has an absolute right to object under Article 21(2) GDPR.

As website publishers often desire to process their visitors’ personal data for purposes beyond what is necessary to serve the website, and beyond what can be based on legitimate interest, a website operator often wants to ask whether the visitor consents to such processing. Such communication currently tends to be done via highly disruptive and repetitive interfaces that are contained in the web page itself (e.g. ‘cookie banners’), rather than through the web browser or other automated channels.

It is the user's choice how to communicate the exercise of GDPR rights to a data controller — the user could send an email, letter, or click a button on a website. In addition, technical means can be used:

- Article 21(5) GDPR expressly provides that *"the data subject may exercise his or her right to object by automated means using technical specifications"*.
- Recital 32 of the GDPR also makes clear that requesting and giving consent could take many forms, which *"could include ticking a box when visiting an internet website, [or] choosing technical settings for information society services […]"*, as long as it satisfies the requirements such as being informed and unambiguous.
- Recital 66 of Directive 2009/136/EC, which updates the 2002 ePrivacy Directive, likewise states that *"the user's consent to processing may be expressed by using the appropriate settings of a browser or other application"*.
- The proposed ePrivacy Regulation (2017/0003 (COD)) equally foresees automated means to communicate data subject preferences.

Despite various legal provisions suggesting its validity, a standardised means for communicating GDPR rights has thus far been lacking.

This specification defines automated means for website visitors to give or refuse consent for the specific purposes that the data controller describes, to withdraw any consent already given, as well as to object to processing for direct marketing purposes based on the data controller's legitimate interest. This enables the user to easily manage data protection decisions through the web browser, and possibly to customise how requests are presented and responded to (e.g. using a browser extension to import lists of trusted websites). The result could be comparable to the way websites ask for permission to access a webcam or microphone: the browser keeps track of the user's decisions on a site-by-site basis, ensures that the user gets a genuinely free choice, and puts the user in control over their decisions.

2. Technical overview §

This section is non-normative.

The protocol described in this document interacts between the website and the user agent (i.e. the browser). The website provides the user agent with a machine-readable "consent requests list" that specifies the data processing purposes for which it requests the user's consent. The user agent responds the user's decisions to the website. The current specification defines two technical paths for these interactions:

1. Between the website *back-end*, i.e. a web server, and the user agent; by means of an HTTP `Link` header (or an equivalent HTML `<link>`) pointing to a JSON file, and the `ADPC` header.
2. Between the website *front-end*, i.e. a web page, and the user agent; by means of a JavaScript interface.

The two methods are equivalent in meaning. There may be technical reasons to use one over the other, or even combine them. For example, the JavaScript approach obviously only works if the user agent supports JavaScript, but it can be used without requiring changes to the back-end infrastructure. Moreover, it permits requesting consent in response to the user's interactions with a page.

The options to present to the user are organised as a list of request strings, each associated with an identifier. In the HTTP-based approach this list is encoded as a JSON file that the website links to. In the JavaScript-based approach, it is passed directly as a JavaScript object to the DOM interface `navigator.dataProtectionControl.request(…)`.

The user's response is presented to the website by listing the identifiers of the requests they consent to. In the HTTP-based approach, this list is sent in the `ADPC` HTTP header in a subsequent HTTP request, while in the JavaScript-based approach, the list is the final return value of the DOM interface.

In the subsequent sections, the communication protocol is defined in detail. First come examples (§ 3. Example) and relevant definitions (§ 4. Definitions). Then § 6. Signals specifies the messages that the two sides exchange, defining the meaning of each. The two sections following it detail how these messages are technically conveyed via either the HTTP-based (§ 7. HTTP-based interaction) or JavaScript-based (§ 8. JavaScript-based interaction) approach.

3. Example §

This section is non-normative.

EXAMPLE 1: Full example, using HTTP-based interaction

The website `example.org` wants to request its visitors' consent to set and access cookies on their device, as well as to create a personalised advertising profile. It describes these activities and purposes in a "consent requests resource":

```
{
  "consentRequests": {
    "cookies": "Store and/or access cookies on your device.",
    "ads_profiling": "Create a personalised ads profile."
  }
}
```

Along with every web page or other resource the website delivers to a visitor, it includes a pointer to this consent requests resource. For example, given a visitor requesting `https://example.org/some/page`:

```
GET /some/page HTTP/1.1
Host: example.org
```

The website may respond with:

```
HTTP/1.1 200 OK
…
Link: </our-consent-requests.json>; rel="consent-requests"
…
```

When the web browser detects this link, it notifies the user that the website would like to request consent. The user could be prompted, perhaps through a pop-up notification, with the purpose descriptions specified by the website. For each purpose, they are given the option to consent to that processing. In this example, we will assume they give consent to set and access cookies, but not for personalised advertising.

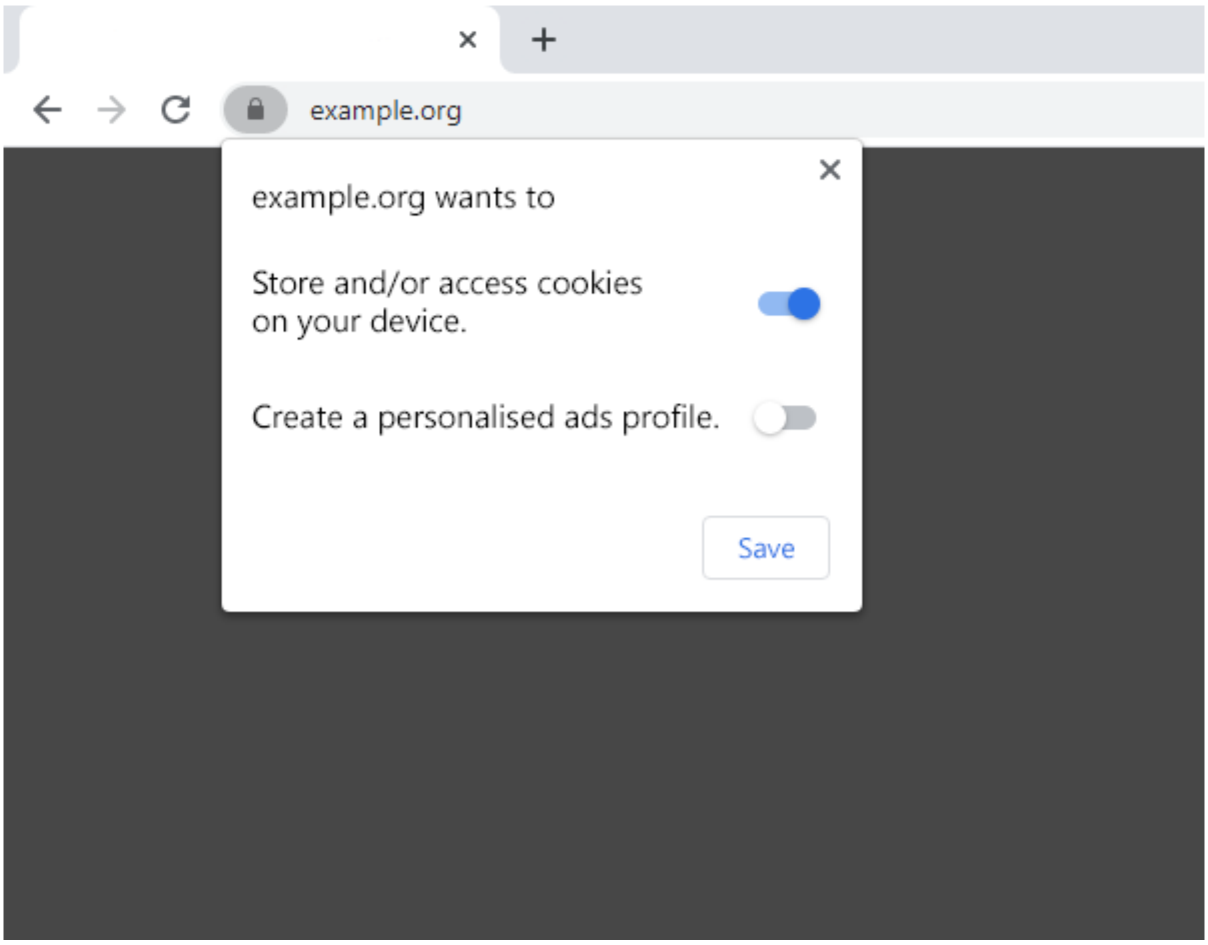
example.org
example.org wants to
Store and/or access cookies on your device.
Create a personalised ads profile.
Save

Figure 1 Possible presentation of a consent request by the web browser

On subsequent requests to `example.org`, the web browser sends along the identifiers of the processing purposes that were consented to (in this case, `cookies`). For the avoidance of doubt, the user might also want to explicitly withdraw consent for any other processing purposes that are not listed here, but that may previously have been given via other means. A next request to website would include the `ADPC` header to convey these signals:

```
GET /some/page HTTP/1.1
Host: example.org
ADPC: withdraw=*, consent=cookies
…
```

Now the website has its answer, and can set and access cookies on the user's device. The web browser may automatically resend this consent decision on subsequent visits to `example.org`.

4. Definitions §

This document touches on data protection regulations as well as technical specifications, which tend to use very different concepts and terminology. In this document, the following terms are used to map data protection concepts into a technical specification:

user
The person visiting or interacting with the website.

This specification uses the word “user” as a term that includes both “data subjects” as defined under Article 4(1) GDPR and “users” as defined in Article 2(a) ePrivacy Directive.

> NOTE
>
> While this specification uses definitions from GDPR and ePrivacy, it may be used in any jurisdiction that operates under similar laws. Future versions of this specification may include references to such laws.

user agent
Any software that retrieves, renders and facilitates end-user interaction with web content. The user may communicate with the controller via the user agent. The term is used interchangeably with “browser”.

controller
The body that provides a website and determines the purposes and means of the processing of personal data or other information stored in the terminal equipment of the user.

This specification uses the word “controller” as a term that includes both the “controller” as defined under Article 4(7) GDPR and the “provider of an information society service” as used in Article 5(3) ePrivacy Directive.

website
The information society service through which the user interacts with a controller. The controller may communicate with the user via the website. A website is delineated by its URL, where any URLs whose origins are schemelessly same site are understood as belonging to the same website.

> EXAMPLE 2
>
> A visit to `https://www.example.org/some/page` and a visit to `http://blog.example.org/another/page` are considered as visiting the same website, that could simply be referred to as `example.org`.

> NOTE
>
> A future version of this specification, or user agents’ implementations of it, could enable treating multiple websites or services by the same data controller (for example, `example.com` and `example.org` or a mobile application) as a combined information society service in the context of data protection control. See the related work in the Privacy Community Group on the First-Party Sets proposal.

5. Scope §

5.1 Personal scope §

The same person may or may not be recognisable to the website on a subsequent visit (for example when the user deletes stored IDs or uses another device or account), and may thus be considered a new user from the website’s perspective.

The scope of the user’s exercise of rights is therefore limited to any personal data and information that relates to the user present in any transaction.

EXAMPLE 3

A website has placed different IDs on the two devices of a user. If the website linked both IDs to the same user account, the user’s exercise of rights must cover transactions from both devices.

If, on the other hand, the website is unable to link both IDs to the same user, the user’s exercise of rights only extends to any personal data or information linked to the ID present in the given transaction.

5.2 Material scope §

The signal expressing the user’s exercise of rights includes any processing of personal data or information based on consent (Article 6(1)(a) GDPR and article 5(3) ePrivacy Directive) or for direct marketing purposes (Article 6(1)(f) and 21(2) GDPR and Article 13(1) ePrivacy Directive).

EXAMPLE 4

When a website legitimately uses personal data for security purposes under Article 6(1)(f) GDPR or to fulfil a contract unter Article 6(1)(b) GDPR, the signal does not apply.

5.3 Territorial scope §

The website may determine the territorial scope where it provides support for this specification. Limited support may be expressed by not including the `Link` header (or the equivalent `<link>` element) in a transaction.

NOTE

The GDPR requires controllers in the EEA to apply the GDPR globally. Non-EEA controllers are usually required to apply the GDPR only for users in the EEA market. Some controllers may use the same system globally, independent of the legal requirements.

6. Signals §

Regardless whether the protocol is used via the HTTP-based or JavaScript-based approach, conceptually the same messages are exchanged between website and user agent. This section describes the messages and their meaning.

6.1 Requesting consent §

The typical communication flow starts with the website requesting its visitor for consent to specific data processing purposes. The website can ***request consent*** from the user for zero or more processing purposes by presenting the user agent a consent requests list.

A ***consent requests list*** is an array containing zero or more consent requests, each representing a processing purpose. A ***consent request*** is an associative array containing the following attributes:

text

The ***request text***: an arbitrary Unicode string that is to be presented to the user. The text should be formulated such that it allows an unambiguous affirmative or negative reaction by the user, such as clicking an “accept” button or ticking a checkbox.

NOTE

While technically the request text may be arbitrary, legally it has to satisfy the requirements under Article 4(7) GDPR, such as allowing a specific and informed consent, using clear, concise and plain language (Recital 42) to enable the website to rely on the affirmative reaction by the user.

id

A ***request identifier***: a shorthand that is used to refer to this consent request. Within the website that makes the consent request, the request identifier *MUST* uniquely correspond to this specific consent request, in order to ensure no ambiguity arises as to which wording of a request the user has consented to. Hence a website must choose a new identifier when e.g. it modifies its request text, perhaps by including a version number as part of the identifier.

Except in standardised consent requests, the identifier consists of an arbitrary string containing uppercase and lowercase latin characters, digits, hyphen (`-`), period (`.`), underscore (`_`), and tilde (`~`) (i.e. any unreserved characters in the URI syntax).

NOTE

Future versions of this specification may define other members, to provide relevant information in a structured, machine-readable way.

EXAMPLE 5: A consent requests list for multiple processing purposes, in JavaScript object notation

```
[
  {
    "id": "q1analytics",
    "text": "We track and analyse your visit(s) on this website, for improving our produ
  },
  {
    "id": "q2recommendation",
    "text": "We observe your interaction with our content to personalise your experience
  },
  {
    "id": "q3advertising",
    "text": "We observe your interaction with our content to personalise advertising to
  },
  {
    "id": "q4thirdPartyAdvertising",
    "text": "We let third party TripleView™ observe your interaction with our content to
  }
]
```

A ***standardised consent request*** is a consent request whose request identifier is a URI rather than an arbitrary string. The attributes of a standardised consent request *MUST* uniquely correspond to the request identifier globally, rather than only within a website.

NOTE

While this standardises the form of the consent request globally, it does of course not imply that a user’s response to any such request must apply to all websites that present the same standardised consent request. The standardisation of consent requests does however open up such possibility for users, as they could for example instruct their agent to refuse particular requests regardless of which website makes it. With or without such blanket responses, it can help reduce information overload and decision fatigue on the side of users, while it may help with compliance efforts on the side of websites.

EXAMPLE 6

An organisation such as the Interactive Advertising Bureau might require all adopters of its “Transparency and Consent Framework” to use standardised requests for the purposes defined in that framework. Such an adopter might then present the following consent requests list:

```
[
  {
    "id": "https://iab.com/tcf2/ns/#purpose1",
    "text": "Cookies, device identifiers, or other information can be stored or accessed
  },
  {
    "id": "https://iab.com/tcf2/ns/#purpose2",
    "text": "Ads can be shown to you based on the content you’re viewing, the app you’re
  },
  {
    "id": "https://iab.com/tcf2/ns/#purpose3",
    "text": "A profile can be built about you and your interests to show you personalise
  },
  {
    "id": "https://iab.com/tcf2/ns/#purpose4",
    "text": "Personalised ads can be shown to you based on a profile about you ads that
  }
]
```

A website *MUST NOT* request consent if it will not respect the user’s decisions. The act of requesting consent thus doubles as an indication of compliance. A website that does not need to request consent, but wishes to (or is obliged to) signal that it supports the protocol, can simply request consent with zero requests.

A website *SHOULD* repeat its consent requests with every page it serves as long as they are applicable. The repetitions help inform the user agent that the consent requests are still relevant. The user agent can recognise which requests the user has already responded to before, and automatically repeat their responses to the website, and can determine which requests to present to the user.

Presenting the requests involves presenting the human-readable request text, along with the option to freely and unambiguously indicate an affirmative or negative wish of the user in line with Article 4(11) GDPR, such as by choosing between an “accept” and “reject” button or by ticking or not ticking an adjacent checkbox.

The exact user agent behaviour may depend on its implementation choices, its interaction modality and user preferences. User agents *MUST* use objective, non-discriminatory rules to determine which consent requests are presented to the user, and in what manner.

EXAMPLE 7

A graphical web browser may show a pop-up, while a voice web browser may read out the consent requests. Either might also first notify the user about the website’s requests before showing the requests themselves, or respect a user’s expressed preference to e.g. only ask for consent after the third visit to any website.

6.2 Giving consent §

To ***give consent*** to zero or more specific processing purposes, the user agent presents the website with a list of the corresponding identifiers.

EXAMPLE 8

From the list in Example 5, assume the user consents to the first two purposes. The user agents passes the list of corresponding identifiers, `q1analytics` and `q2recommendation`. In the HTTP approach this could be expressed by sending this header:

```
ADPC: consent="q1analytics q2recommendation"
```

If the user would have refused to give consent to any of the requests, the list would simply be empty.

For legal validity, the user agent *MUST NOT* give consent without properly presenting the requests to the user and without freely given, specific, informed and unambiguous affirmative indication by the user. The user agent can remember the user’s previously given consent and repeat it on subsequent repetitions of the same request.

6.3 Withdrawing consent §

The user may ***withdraw consent*** that was previously given; the consent is represented by the identifier that was used to request it.

The user can also ***withdraw all consent*** for purposes not explicitly consented to in the current exchange. This also withdraws consent that was given ‘out of band’ (i.e. not through this protocol), and can therefore be useful to ensure that the user has a complete overview and control over the processing purposes they consented to.

EXAMPLE 9

After having previously given consent as described in Example 8, assume the user expresses they want to withdraw their consent to data processing for the purpose of analytics, possibly by unchecking the corresponding checkbox in their browser. The user agent passes a list containing the corresponding identifier, `q1analytics`. In the HTTP approach this could be expressed by sending this header:

```
ADPC: withdraw=q1analytics
```

If the user would wish to withdraw all previously given consents, the user agent would send the following header:

```
ADPC: withdraw=*
```

Note that as consent has to be given to *specific* purposes, there is no equivalent `consent=*`.

The user agent *MUST* make it as easy to withdraw as to give consent in order to comply with Article 7(3) GDPR. This requires that the option to withdraw consent via the user agent must be as easy to access and exercise as the option to consent to the processing operation.

6.4 Objecting to processing §

The user may ***object to processing of their personal data*** as provided for under Article 21 GDPR. Objection involves passing the appropriate objection identifier.

An ***objection identifier*** is a string corresponding to a type of objection. This specification defines only one objection identifier: `direct-marketing`. The user may provide this identifier to object to processing of their personal data for direct marketing purposes, as provided for under Article 21(2) GDPR.

NOTE

At time of writing, there appears to no general agreement about the scope of “direct marketing” in Article 21(2) GDPR. This specification does not take any position on the scope, but merely allows users to object to whatever falls within this scope.

NOTE

While Article 21(1) and 21(6) GDPR also define a right to object for other purposes than direct marketing, objection under these paragraphs would usually require an exchange about the specific situation of the user, which seems to require more specific interactions.

Other specifications, or future versions of this specification, could define other types of objections, possibly based on other data protection laws in other jurisdictions.

6.5 Combining signals §

To ensure that the records of user preferences stored by the user agent and by the website stay synchronised and to ensure that the signalled preferences prevail over other interactions, it may be useful to combine multiple signals.

When signals are combined, specific signals (such as giving consent for specific processing purposes) *SHALL* prevail over more general signals (such as withdrawing all consent).

EXAMPLE 10

A user may instruct the user agent to generally withdraw all previously given consent as well as to object to any direct marketing, but consent to the purposes identified by `q1analytics` and `q2recommendation` as in Example 8. In the HTTP approach this could be expressed by sending this header:

```
ADPC: withdraw=*, object=direct-marketing, consent="q1analytics q2recommendation"
```

The website would then know that generally no processing based on consent (Article 6(1)(a) GDPR and/or Article 5(3) ePrivacy Directive) and no processing for direct marketing (Article 21(2) GDPR) is allowed, but that the user did consent to the purposes identified by `q1analytics` and `q2recommendation`. Any consent that was previously obtained (including via e.g. a previously used consent banner) would now be understood to be withdrawn. This ensures the user that *only* those consents they can view and control through their browser are valid.

7. HTTP-based interaction §

This section defines the first of the two ways to use the ADPC mechanism, which primarily communicates using the HTTP headers exchanged between the web server and user agent, while using a JSON resource to convey the consent requests.

NOTE

An individual HTTP interaction consists of a request by the user, and a response from the web server. However, the communication flow for consent tends to require the reverse: the website informs the user about its data processing purposes, to which the visitor may respond by giving consent, or refusing to. Therefore, the ADPC flow starts with the HTTP response of the server, to which the user agent responds in subsequent HTTP request(s). Because the user agent would normally only send the ADPC header if the server proclaims to support the protocol, this approach should also limit the options for device fingerprinting.

7.1 Requesting consent §

A website lists the processing purposes it requests consent for in a ***consent requests resource***, which is a JSON file containing an object with the following attributes:

consentRequests
A consent requests list.

EXAMPLE 11: A consent requests resource

```
{
  "consentRequests": [
    {
      "id": "q1analytics",
      "text": "We track and analyse your visit(s) on this website, for improving our pro
    },
    {
      "id": "q2recommendation",
      "text": "We observe your interaction with our content to personalise your experien
    },
  ]
}
```

To request consent, the website points to such a consent requests resource using the HTTP `Link` header, with the relation type `consent-requests`.

EXAMPLE 12: Pointing to the consent requests resource via an HTTP Link header

```
GET /some/page HTTP/1.1

HTTP/1.1 200 OK
Link: </our-consent-requests.json>; rel="consent-requests"
…
```

When returning an HTML or XML document, instead of adding the `Link` header, an equivalent `<link>` element can be added to the document, with the same semantics as the header.

EXAMPLE 13: Pointing to the consent requests resource via an HTML <link> element

```
<!doctype html>
<html>
  <head>
    <link href="/our-consent-requests.json" rel="consent-requests" />
    …
```

NOTE

Because the consent requests resource is linked to, rather than directly included in the response, the traffic overhead remains small. Caching mechanisms (e.g. using the `ETag` and `Cache-Control` headers) can avoid the repeated transfer of the consent requests resource.

EXAMPLE 14: Consent request resources in multiple languages

Using the standard content negotiation procedure, the user agent and web server would use the `Accept-Language` and `Content-Language` headers to obtain the consent requests resource in the user's preferred language.

The link's `hreflang` attribute can be used to hint at the language(s) in which the consent requests resource is available:

```
GET /some/page HTTP/1.1


HTTP/1.1 200 OK
Link: </our-consent-requests.json>; rel="consent-requests"; hreflang="en fr de"
…
```

Alternatively, a website could give each language its own URL, and declare multiple links with different `hreflang` values to point to translations at different URLs:

```
GET /some/page HTTP/1.1


HTTP/1.1 200 OK
Link: </our-dp-policy.en.json>; rel="consent-requests"; hreflang="en"
Link: </our-dp-policy.fr.json>; rel="consent-requests"; hreflang="fr"
…
```

The different translations of the consent requests would use the same identifiers.

When the user agent detects a `consent-requests` link in a document in the top-level browsing context, it would typically fetch and parse the linked JSON resource and, if this succeeds, present the requests to the user. It *MAY* reduce traffic by only fetching the resource once the user decides to interact with the consent requests.

7.1.1 Making zero requests §

A website may want to explicitly express that it does not request any consent. The obvious approach would be to present a consent requests resource with an empty object as the value of `consentRequests`. In the HTTP-based approach, this could however cause an unnecessary round trip for obtaining this consent requests resource. Moreover, in a user agent that notifies or hints the user about consent requests before it retrieves the consent requests resource, it could cause a confusing experience.

Instead of presenting a consent requests resource without consent requests, a website *SHOULD* therefore link to the special target `about:blank`.

EXAMPLE 15: Supporting ADPC, not requesting any consent

```
GET /some/page HTTP/1.1


HTTP/1.1 200 OK
Link: <about:blank>; rel="consent-requests"
…
```

7.2 Giving and withdrawing consent §

To give consent or withdraw consent to zero or more specific processing purposes listed in the received consent requests resource, the user agent adds the `ADPC` HTTP header in its subsequent HTTP requests to the website.

The value of the `ADPC` header is set to the characters `consent=`, or `withdraw=`, respectively, followed by a double-quoted string containing the corresponding request identifiers, separated by spaces. If the list has only one identifier, the double quotes can be omitted. If it has zero identifiers, the value can equivalently be empty, or the header can be omitted altogether.

To convey multiple decisions, the `ADPC` header can be used any number of times in a single HTTP request, in arbitrary order. As is standard with HTTP headers, multiple header values can either be listed as separate headers, or be concatenated with a comma (optionally surrounded by whitespace).

The user agent *SHOULD* repeat the ADPC header with every HTTP request it makes to the website, as long as it is applicable. The repetitions enable a website to know the user's decision without keeping records itself. The user agent *MAY* send a stand-alone HTTP request to ensure a user's decision is conveyed as soon as possible, instead of waiting for the next natural occasion to transmit the signal.

EXAMPLE 16: Giving consent via HTTP

To the consent requests presented in Example 5, assume the user has decided to consent to the processing of their personal data for the purposes of product improvement and personalised recommendations. On the subsequent request(s) to the server, the user agent adds the `ADPC` header:

```
GET /some/page HTTP/1.1
ADPC: consent="q1analytics q2recommendation"
…
```

EXAMPLE 17: Withdrawing consent via HTTP

At a subsequent moment, assume the user wishes to withdraw the consent to the personal data processing for the purpose of content recommendation. For clarity, the consent for the purpose of product usage analytics may be repeated:

```
GET /some/page HTTP/1.1
ADPC: withdraw=q2recommendation, consent=q1analytics
…
```

To withdraw all consent, the special value `*` can be used in place of an identifier.

EXAMPLE 18: Withdrawing all consent via HTTP

Assume the user wishes to withdraw all consent it has previously given a website:

```
GET /some/page HTTP/1.1
ADPC: withdraw=*
…
```

Or, the user may wish to consent to particular processing purposes, but withdraw any other consent they may have given in the past:

```
GET /some/page HTTP/1.1
ADPC: withdraw=*, consent="q1analytics q2recommendation"
…
```

A ***stand-alone HTTP request*** is an HTTP `HEAD` request to an arbitrary path on the website's origin. It can be used to deliver HTTP headers without requesting any resource.

EXAMPLE 19: A stand-alone HTTP request to withdraw all consent

A user can find, locally in their browser, an overview of previously given consent, and may wish to withdraw all consent for a chosen website even while not visiting it at that moment. The browser sends the appropriate `ADPC` header along with a stand-alone HTTP request to an arbitrary path of the website.

```
HEAD / HTTP/1.1
ADPC: withdraw=*
…
```

7.3 Objecting to processing §

To object to processing of their personal data, the user agent adds the `ADPC` HTTP header to any HTTP request to the website, with the value `object=` followed by a double-quoted string containing zero or more objection identifiers. If the list has only one identifier, the double quotes can be omitted. If it has zero identifiers, the value can equivalently be empty, or the header can be omitted altogether.

EXAMPLE 20: An HTTP request to object to direct marketing

The user expresses their objection to the processing of their personal data for the purpose of direct marketing, along with a request to a web page:

```
GET /some/page HTTP/1.1
ADPC: object=direct-marketing
…
```

Extending Example 19, the user expresses the objection in combination with a withdrawal of all previous consent:

```
GET /some/page HTTP/1.1
ADPC: withdraw=*, object=direct-marketing
…
```

8. JavaScript-based interaction §

While the HTTP signalling approach can be sufficient, there are several reasons why a website may prefer to communicate in other ways. For example:

- Statically hosted web content that cannot adapt to the HTTP signals.
- Third party services or proxies are used that do not give access to the HTTP headers.
- Scripts or other content involved in data processing is loaded from third parties, which will therefore not receive the user’s decisions.

EXAMPLE 21

Assume `website.example` uses a third-party CMP, loaded directly from `cmp.example` using a `<script>` tag:

```
<script src="https://cmp.example/embed.js?siteid=website.example"/>
```

While the `website.example` web server would receive the user’s consent decisions through the HTTP request headers, the `cmp.example` web server would not. Hence, `cmp.example` cannot adjust the script’s content to comply with the user’s decisions. The script will be executed in context of the `website.example` page, but cannot read the HTTP headers that had been sent while requesting that page. The JavaScript interface provides a practical solution to this issue.

8.1 Requesting consent §

To request consent for zero or more processing purposes, a website’s scripts can invoke the `request()` function, passing it a consent requests list. The user agent will then present the requests to the user, or/and directly return a response based on the user’s previous responses or preferences.

Requesting consent can only be done by the web page the user consciously visits, and not by e.g. resources embedded within it. Therefore, if the `request()` function was invoked by a script that is not running in the top-level browsing context, the user agent *MUST NOT* act upon the consent request, and simply resolve with an empty user decisions object.

EXAMPLE 22: Requesting consent for several data processing purposes in JavaScript

```
const userDecisionsPromise = navigator.dataProtectionControl.request([
  {
    "id": "q1analytics",
    "text": "We track and analyse your visit(s) on this website, for improving our produ
  },
  {
    "id": "q2recommendation",
    "text": "We observe your interaction with our content to personalise your experience
  },
  {
    "id": "q3advertising",
    "text": "We observe your interaction with our content to personalise advertising to
  },
  {
    "id": "q4thirdPartyAdvertising",
    "text": "We let third party TripleView™ observe your interaction with our content to
  }
]);
```

8.2 Giving and withdrawing consent, and objecting to processing §

To signal user decisions to the website, the user agent passes the appropriate values in the return value of the `request()` function. The return value is a `Promise` that resolves with a user decisions object.

Whenever the user changes their decisions relating to the website while visiting it, the user agent notifies the page by triggering the `decisionchange` event on `dataProtectionControl`, which includes the updated user decisions object.

A ***user decisions object*** is a JavaScript object, the members of which signal the user’s decisions to the website. Each member is optional:

consent
: An array containing any request identifiers for which the user gives consent.

withdraw
: An array containing any request identifiers for which the user withdraws consent, and/or possibly containing the string `"*"` to withdraw all consent.

object
: An array containing any objection identifiers that indicate the user objects to processing of their personal data.

EXAMPLE 23: Giving consent via JavaScript

Suppose that, in response to the request shown in Example 22, the user decided to consent to the processing of their personal data for the purposes of product improvement and personalised recommendations. The value of `userDecisionsPromise` would then resolve to:

```
{
  consent: ["q1analytics", "q2recommendation"]
}
```

Unlike with the HTTP approach, where the user agent can send request headers, in JavaScript there is not an obvious way for the user agent to take the initiative to send a signal to the website. In cases where a website does not or cannot read the ADPC HTTP header, it *MUST* on every page visit invoke and await the return value of `request()`, as well as listen to the `decisionchange` event, in order to ensure it becomes aware of a user's decisions.

EXAMPLE 24: Withdrawing consent via JavaScript

If, at any moment, the user decides to withdraw all consent for the website, and object to data processing for the purpose of direct marketing, the next invocation of `request()` might resolve to the following value:

```
{
  consent: [],
  withdraw: ["*"],
  object: ["direct-marketing"]
}
```

8.3 Interface definition §

WebIDL

```
[Exposed=Navigator]
interface DataProtectionControl : EventTarget {
  Promise<UserDecisionsObject> request(object consentRequestsList);
};

[Exposed=DataProtectionControl]
interface AdpcEvent {
  readonly attribute UserDecisionsObject userDecisions;
};

[Exposed=DataProtectionControl]
interface UserDecisionsObject {
  readonly attribute DOMStringList? consent;
  readonly attribute DOMStringList? withdraw;
  readonly attribute DOMStringList? _object;
};

partial interface Navigator {
  [SameObject] readonly attribute DataProtectionControl dataProtectionControl;
};
```

The ***dataProtectionControl*** interface enables a web page to request consent from the user and learn about their data protection decisions.

The ***request()*** method can be used to request consent, as described in § 8.1 Requesting consent.

EDITOR'S NOTE

The exact interface definition is yet to be completed.

NOTE: Relation with the Permissions specification

While it remains close to the equivalent HTTP header signals described in § 7. HTTP-based interaction, aspects of this JavaScript interface were inspired by the Permissions draft specification, and trying to make the two APIs more similar could be helpful for web developers.

Moreover, closer integration of these specifications may be worth consideration; this way, for example, a prompt could request the user both for technical access to one's location data, and for consent to use this data for a given purpose.

9. Bulk consent requests lists §

9.1 Concept §

This section is non-normative.

Users regularly feel overwhelmed by consent requests ('consent fatigue'). Enabeling trusted parties to suggest including or excluding certain consent requests could allow users and user agents to determine which consent requests to prioritise or automatically consent to, and which consent requests to ignore or automatically reject. Websites, groups of controllers or consumer groups may promote inclusion lists, for example of websites providing quality journalism or for the purpose of informing users about regular discounts.

The user agent may allow to import such third party bulk consent requests lists, which may be made available as a file or as a continuously updated online resource. The discovery and exchange mechanisms used are beyond the scope of this specification.

The exact user agent behaviour after importing such a list may depend on its implementation choices, its interaction modality and user preferences. For example, excluded consent requests might only be shown when a user has visited a website more often, while included consent requests are shown instantly. Other implementations might automatically consent to all included consent requests and automatically reject all excluded consent requests.

If a user agent allows to automatically consent to included consent requests, it *MUST* ensure a freely given, specific, informed and unambiguous indication of the user's wishes for each consent request.

9.2 Definition §

A ***bulk consent requests list*** is a JSON file containing an object with the following attributes:

include
An array of consent request descriptors that should be included in the users' considerations.

exclude
An array of consent request descriptors that should be excluded in the users' considerations.

A ***consent request descriptor*** is an object with the following members:

website
A string containing the registrable domain of the website, or an asterisk (*) if it applies to any website.

consentRequests
An array of consent requests that should be included or excluded.

EXAMPLE 25: A bulk consent requests list

```
{
  "include": [
    {
      "website": "example.org",
      "consentRequests": [
        {
          "id": "q1analytics",
          "text": "We track and analyse your visit(s) on this website, for improving our
        },
        {
          "id": "q2recommendation",
          "text": "We observe your interaction with our content to personalise your expe
        }
      ]
    },
    {
      "website: "*",
      "consentRequests": [
        {
          "id": "https://iab.com/tcf2/ns/#purpose1",
          "text": "Cookies, device identifiers, or other information can be stored or ac
        },
        {
          "id": "https://iab.com/tcf2/ns/#purpose2",
          "text": "Ads can be shown to you based on the content you're viewing, the app
        }
      ]
    }
  ],
  "exclude": [
    {
      "website": "*"
      "consentRequests": [
        {
          "id": "https://iab.com/tcf2/ns/#purpose3",
          "text": "A profile can be built about you and your interests to show you perso
        },
        {
          "id": "https://iab.com/tcf2/ns/#purpose4",
          "text": "Personalised ads can be shown to you based on a profile about you ads
        }
      ]
    }
  ]
}
```

10. Compatibility considerations §

This section is non-normative.

Users may use various forms of communicating consent, withdrawal of consent, or objections — the user could send an email, letter, or click a button on a website. Independent of the communication channel, the most recent communication would usually override the previous exercise of rights. As the ADPC signal would typically be communicated in every interaction with a website, it would quickly override previous expressions through any other communication, like consent banners, emails or letters.

If the ADPC signal is sent in the same transaction as another signal with related meaning (e.g. when clicking an “agree” button on a website, or sending another signal such as a DNT or Sec-GPC HTTP header), any non-contradicting communication can be interpreted combinedly without problem. Any expressions of consent that are in conflict with each other will not be “unambiguous” as required by Article 4(7) GDPR, and should thus be interpreted as a lack of valid consent.

11. Privacy considerations §

This section is non-normative.

While the primary purpose of the specified mechanism is to help improve personal data protection, it is important to recognise that the approach is in essence legal, rather than technical. The mechanism conveys users’ decisions in a machine-readable manner, which the website might be legally obliged to respect, but the effective protection relies on the website’s compliance with the law. Privacy impact considerations can therefore be divided into the potential benefits from its use, and potential harms from its abuse.

11.1 Privacy impact in case of compliant websites §

To assess the impact, we compare the adoption of the specified mechanism with the commonly observed alternative: requests for consent via interfaces contained within the website’s pages, and stored using cookies or other browser storage. Adoption of this specification could yield the following benefits for user privacy:

- The user can reject or remove cookies in their browser, or use a ‘private browsing mode’, without being presented with a consent banner on every website they revisit. Removing cookies can greatly improve user’s privacy, but has become unattractive since the advent of cookie-based consent management systems.
- The user can control data protection decisions for multiple websites in aggregate. For example, they can review the consents they have given to multiple parties, and possibly withdraw many or all of them at once.
- Because the interactions of requesting and responding are both machine-readable and standardised, a range of possibilities open for more user-centric design. For example, the user agent can offer customised, individualised behaviour for those with special needs or preferences, or help reduce information overload by blocking excessive consent requests.
- The user agent controls the interaction, ensuring that each request looks and behaves similarly. For example, the accept and reject buttons are always presented in the same order, avoiding confusion and unintended responses. This also reduces the ability for websites to intentionally create such confusion (known as ‘dark patterns’).

11.2 Privacy impact in case of non-compliant websites §

Even if the mechanism benefits privacy in websites that abide by it, it would be undesirable if it can harm their privacy in cases where websites do not comply. This section discusses limitations of the specified mechanism and some mitigations.

11.2.1 Misplaced trust §

First of all, this mechanism cannot prevent websites from giving false or incomplete information, or simply disrespecting the user's decisions. A false pretense of control may erode trust in the system. While this might equally be the case without use of this mechanism, the presentation through the web browser interface, which is generally more trusted than the website being visited, may give a false sense that decisions are enforced by the user agent, as is the case with permission requests for e.g. microphone access.

11.2.2 Tracking §

A common concern with a new web standard is whether it enables websites to track users. Because the specified mechanism is only used with web pages in the top-level browsing context, and the user decisions are only presented to the individual website they apply to, it does not introduce new vectors for *cross-website tracking*. The specified HTTP headers are not passed along with, nor read from transactions with, a web page's subresources, and the JavaScript interface is unusable inside framed pages.

However, a limited ability to do *first-party tracking* is unavoidable given that users express their decisions, which will necessarily convey some information. The user's data protection decisions, simply by being different from those of other persons, could be used to help re-identify them on subsequent visits.

The situation here is similar to that of first-party cookies, although it is made less impactful because the requests are visible to the user, and the responses are made by the user rather than set arbitrarily by the website. Moreover, the entropy of user decisions is likely very low: if a website asks four consent questions, these provide at most four bits of information, but in practice much less because users do not choose their responses perfectly at random. Especially if a website makes, say, fourty consent requests, users are unlikely to make fourty independent decisions: rejecting or accepting all requests at once is a common response.

Besides the individual users' responses, without further precautions the request identifiers also risk to be usable as persistent tracking vectors. A malicious website could, rather than having a static list of consent requests, customise the request identifiers for each user to recognise the user again (if they consented) during a subsequent visit. Various approaches could help prevent this form of tracking. For example, user agents could refrain from transmitting the consent header value along with the first HTTP request to a website in a new session, in order to first verify whether the website still makes the same requests as before.

Even though the mechanism does not enable cross-website tracking, and is less impactful than first-party cookies, the possibility to track users would need to be much less than with cookies, so that users can trust they keep their data protection decisions when removing their cookies. To this end, mitigations should be developed, and implementers should evaluate their abilities to limit entropy and may make trade-offs between efficiency and anonimity.

11.2.3 Third-party scripts §

Another potential privacy/security risk arises from the ability of a third-party script loaded into the web page to use the JavaScript interface as if it was part of the page itself. It could make the web page request consent and observe the user's decisions for the website, and possibly transmit information back to its creators or other parties. However, this specification may not significantly exacerbate this already existing issue: any included third-party script needs to be fully trusted, and can do worse things than requesting consent. Common security features, such as *Content Security Policy Level 2* and *Subresource Integrity*, can somewhat reduce the risk of including third-party scripts.

11.3 User agent's role in data protection §

The specified mechanism gives the user agent an important role in the exercise of people's data protection rights, and thereby also responsibilities. Following the principle of 'privacy by default', the mechanism is designed to err on the side of less processing when needed. For example, if a step in the protocol is hampered due to the consent requests resource being invalid or temporarily unavailable, the result is that no consent is requested, nor given.

Apart from some basic requirements to e.g. avoid invalid consent, user agents have significant freedom in the implementation of their side of the mechanism. This freedom can be used to further improve people's data protection control, for example by supporting the import of bulk consent requests lists.

While the above analysis covers the case of non-compliant websites, it assumes that user agents are indeed acting, as the term implies, on behalf and in the best interest of the user. While the user in theory has freedom to choose and even customise their user agent, this assumption may often be hampered in practice. User agents could for example be prone to apply 'dark patterns' or unfairly discriminate between websites, due to misaligned interest of its developer. Legal compliance may therefore be relevant for the user agent as well as the website, and wide customisability of user agents through plug-ins/extensions can be an important factor for putting the user in control.

12. 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*, *SHALL*, and *SHOULD* in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

A. Acknowledgements §

The authors are grateful for the contributions and feedback by Alan Dahi, Rob van Eijk, Stefanie Alice Hofer, Horst Kapfenberger, Mandan Kazzazi, Gustaf Neumann, Mike O'Neill, Harshvardhan J. Pandit, Monika Riegler, Stefano Rossetti, and our other colleagues from different institutions around the globe.

This work is partially supported by the Internet Foundation Austria (IPA) within the NetIdee call (RESPECTeD Project; Grant#prj4625).

B. References §

B.1 Normative references §

[dom]
DOM Standard. Anne van Kesteren. WHATWG. Living Standard. URL: https://dom.spec.whatwg.org/

[html]
HTML Standard. Anne van Kesteren; Domenic Denicola; Ian Hickson; Philip Jägenstedt; Simon Pieters. WHATWG. Living Standard. URL: https://html.spec.whatwg.org/multipage/

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

[RFC8174]
Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words. B. Leiba. IETF. May 2017. Best Current

Practice. URL: https://datatracker.ietf.org/doc/html/rfc8174

[URI]
Uniform Resource Identifier (URI): Generic Syntax. T. Berners-Lee; R. Fielding; L. Masinter. IETF. January 2005. Internet Standard. URL: https://datatracker.ietf.org/doc/html/rfc3986

[url]
URL Standard. Anne van Kesteren. WHATWG. Living Standard. URL: https://url.spec.whatwg.org/

[WebIDL]
Web IDL. Boris Zbarsky. W3C. 15 December 2016. W3C Editor's Draft. URL: https://heycam.github.io/webidl/

B.2 Informative references §

[CSP2]
Content Security Policy Level 2. Mike West; Adam Barth; Daniel Veditz. W3C. 15 December 2016. W3C Recommendation. URL: https://www.w3.org/TR/CSP2/

[Permissions]
Permissions. Mounir Lamouri; Marcos Caceres; Jeffrey Yasskin. W3C. 15 June 2021. W3C Working Draft. URL: https://www.w3.org/TR/permissions/

[SRI]
Subresource Integrity. Devdatta Akhawe; Frederik Braun; Francois Marier; Joel Weinberger. W3C. 23 June 2016. W3C Recommendation. URL: https://www.w3.org/TR/SRI/

[tracking-dnt]
Tracking Preference Expression (DNT). Roy Fielding; David Singer. W3C. 17 January 2019. W3C Note. URL: https://www.w3.org/TR/tracking-dnt/

↥