Table of Contents
Introduction
License
Disclaimer
About IAB Tech Lab
About IAB CCPA Compliance Framework
   Final v1.0 Specifications
Relevant Documents
Version History:
Requirements for U.S. Privacy User Signal API
What are you required to support?
What baseline functionality is required?
Where should the string be stored?
How is the API exposed?
   getUSPData
In-app support
   How can vendors that use iframes call the API from an iframe?
      Via SafeFrames
      Without SafeFrames, using postMessage

Introduction
This document outlines technical mechanisms to support communication of U.S. Privacy signal. These signals contain information about disclosures made and choices selected by a user regarding consumer data privacy under U.S. Privacy regulation, and are documented in a separate U.S. Privacy String specification. Version 1 of this U.S. Privacy User Signal API specification only supports signals pertaining to the California Consumer Privacy Act (CCPA).

This specification was created because Digital Properties need a scalable way to establish and persist U.S. Privacy signals. Additionally, downstream vendors need a reliable way to access U.S. Privacy signals when running within a Digital Property's website or app.

This document specifies a lightweight API that may be implemented by Digital Properties for web and mobile in-app to represent U.S. Privacy signals.
License

U.S. Privacy String and API technical specifications governed by the IAB Tech Lab is licensed under a Creative Commons Attribution 3.0 License. To view a copy of this license, visit creativecommons.org/licenses/by/3.0/ or write to Creative Commons, 171 Second Street, Suite 300, San Francisco, CA 94105, USA.

Disclaimer

THE STANDARDS, THE SPECIFICATIONS, THE MEASUREMENT GUIDELINES, AND ANY OTHER MATERIALS OR SERVICES PROVIDED TO OR USED BY YOU HEREUNDER (THE “PRODUCTS AND SERVICES”) ARE PROVIDED “AS IS” AND “AS AVAILABLE,” AND IAB TECHNOLOGY LABORATORY, INC. (“TECH LAB”) MAKES NO WARRANTY WITH RESPECT TO THE SAME AND HEREBY DISCLAIMS ANY AND ALL EXPRESS, IMPLIED, OR STATUTORY WARRANTIES, INCLUDING, WITHOUT LIMITATION, ANY WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, AVAILABILITY, ERROR-FREE OR UNINTERRUPTED OPERATION, AND ANY WARRANTIES ARISING FROM A COURSE OF DEALING, COURSE OF PERFORMANCE, OR USAGE OF TRADE. TO THE EXTENT THAT TECH LAB MAY NOT AS A MATTER OF APPLICABLE LAW DISCLAIM ANY IMPLIED WARRANTY, THE SCOPE AND DURATION OF SUCH WARRANTY WILL BE THE MINIMUM PERMITTED UNDER SUCH LAW. THE PRODUCTS AND SERVICES DO NOT CONSTITUTE BUSINESS OR LEGAL ADVICE. TECH LAB DOES NOT WARRANT THAT THE PRODUCTS AND SERVICES PROVIDED TO OR USED BY YOU HEREUNDER SHALL CAUSE YOU AND/OR YOUR PRODUCTS OR SERVICES TO BE IN COMPLIANCE WITH ANY APPLICABLE LAWS, REGULATIONS, OR SELF-REGULATORY FRAMEWORKS, AND YOU ARE SOLELY RESPONSIBLE FOR COMPLIANCE WITH THE SAME.

About IAB Tech Lab

The IAB Technology Laboratory (Tech Lab) is a non-profit consortium that engages a member community globally to develop foundational technology and standards that enable growth and trust in the digital media ecosystem. Comprised of digital publishers, ad technology firms, agencies, marketers, and other member companies, IAB Tech Lab focuses on improving the digital advertising supply chain, measurement, and consumer experiences, while promoting responsible use of data. Its work includes the OpenRTB real-time bidding protocol, ads.txt anti-fraud specification, Open Measurement SDK for viewability and verification, VAST video specification, and DigiTrust identity service. Board members include ExtremeReach, Facebook, Google, GroupM, Hearst Digital Media, Index Exchange, Integral Ad Science, LinkedIn, LiveRamp, MediaMath, Microsoft, Oracle Data Cloud, Pandora, PubMatic, Quantcast, Rakuten Marketing, Telaria, The Trade Desk, Verizon Media Group, Xandr, and Yahoo! Japan. Established in 2014, the IAB Tech Lab is headquartered in New York City with staff in San Francisco, Seattle, and London. Learn more at https://www.iabtechlab.com.
About IAB CCPA Compliance Framework

The IAB CCPA Compliance Framework comprises of policy and technical work to support CCPA compliance. This document is the work product of the IAB Tech Lab's CCPA/U.S. Privacy Technical Working Group. Policy requirements were developed by a legal affairs group at IAB in the US. The technical specifications documents refer to the guidance within IAB CCPA Compliance Framework Policies.

More information about the Framework is available at iab.com/guidelines/ccpa-framework and technical specifications are available at iabtechlab.com/ccpa.

Final v1.0 Specifications

Final v1.0 technical specifications for the IAB CCPA Compliance Framework may be re-released when the Limited Service Provider Agreement is released to the industry. Furthermore, in the future, the technical component of the Framework may support the communication of data deletion requests.

Relevant Documents

IAB CCPA Compliance Framework Policies
IAB Tech Lab U.S. Privacy String
IAB Tech Lab U.S. Privacy User Signal API
IAB Tech Lab U.S. Privacy OpenRTB Extension

Version History:

<table>
<thead>
<tr>
<th>Date</th>
<th>Version</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>October 2019</td>
<td>1.0</td>
<td>Draft for public comment. Version 1 ONLY supports CCPA Compliance.</td>
</tr>
<tr>
<td>November 18 2019</td>
<td>1.0</td>
<td>Published final public version. Added SafeFrame, iFrame support.</td>
</tr>
<tr>
<td>November 20 2019</td>
<td>1.0</td>
<td>Fixed error referencing in-app key. Correct key is “IABUSPrivacy_String”</td>
</tr>
</tbody>
</table>
Requirements for U.S. Privacy User Signal API

The U.S. Privacy Signal component follows design patterns found in similar privacy compliance frameworks. The design pattern includes how the component is loaded into web pages or native apps and how vendors interact with the USP API. The USP component shall be loaded onto a Digital Property's site or app.

What are you required to support?

To support sending and receiving of the U.S. Privacy String within the User Signal Mechanism, the following functionalities are required:

- Desktop JavaScript API support
- Mobile local storage support, and An API (optionally Swift or Java, etc)
- Macro support (see U.S. Privacy String specification for details)

What baseline functionality is required?

As a baseline, Digital Properties must create a string and make it available to vendors via this API. This string indicates that CCPA does not apply, or signals whether the explicit notice has been provided and the user opted out.

Where should the string be stored?

The Digital Property is responsible for storing the string. It’s recommended to store the string into a 1st party cookie, named “usprivacy”, to and from where the library can write/read it. In case storing on a 1st party cookie is not possible or practical (e.g. on mobile native or if cookies are disabled), a different storage method can be adopted. The API provides individual methods to modify the value of each different section of the string.

How is the API exposed?

The following API function can be provided;

```javascript
__uspapi(Command, Version, Callback)
```
__uspapi() must always be a function at all times, even at initialization – the API must be able to handle calls at all times.

Secondarily, the implementation must provide a proxy for postMessage events targeted to the __uspapi interface sent from within nested iframes. See guidance in this specification here.

At the minimum, the implementation must support the following API commands: 'getUSPData'.

getUSPData

<table>
<thead>
<tr>
<th>argument name</th>
<th>type</th>
<th>optional</th>
<th>value</th>
</tr>
</thead>
<tbody>
<tr>
<td>command</td>
<td>string</td>
<td></td>
<td>'getUSPData'</td>
</tr>
<tr>
<td>version</td>
<td>number</td>
<td></td>
<td>U.S. Privacy spec version</td>
</tr>
<tr>
<td>callback</td>
<td>function</td>
<td></td>
<td>function(uspData: uspdata, success: boolean)</td>
</tr>
</tbody>
</table>

Example:

```javascript
__uspapi('getUSPData', 1, (uspData, success) => {

    if (success) {

        // do something with uspData

    } else {

        // do something else

    }
```

iabtechlab.com/ccpa
If U.S. Privacy does not apply to this user in this context then the string in uspData object will contain “1---”.

The callback shall be called immediately and without any asynchronous logic with whatever information is available in the current state of the library.

A value of false will be passed as the argument to the success callback when no uspData object could be returned.

The callback shall be invoked only once per api call with this command.

**uspData Object**

```json
{
    "version": 1, /* number indicating the U.S. Privacy spec version */
    "uspString": "1YNN", /* string; not applicable: "1---" */
    /* number; 1 applies, 0 doesn’t apply, -1 not set */
}
```

**In-app support**

The encoded string and any related information must be stored on `NSUserDefaults` (iOS) or `SharedPreferences` (Android). This allows:

- Vendors to easily access the string information when they need to;
- The string and any related information to be persisted across app sessions;
- Pre-parsing of the string to enable all typical use-cases, with flexibility to act according to the user’s choices.
Here is the list of the key value pairs provided:

<table>
<thead>
<tr>
<th>Key / Field</th>
<th>Scope</th>
<th>Values</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>IABUSPrivacy_String</td>
<td>optional</td>
<td>String</td>
<td>E.g. “1YNN” Aligns with IAB OpenRTB CCPA Advisory. The String encodes all choices and information.</td>
</tr>
</tbody>
</table>

How can vendors that use iframes call the API from an iframe?

There are two ways to request the US Privacy String from a parent or ancestor’s frame: Via SafeFrames and without SafeFrames, using postMessage.

**Via SafeFrames**

When SafeFrames are used to proxy API requests no changes are required for the USP API or the vendor. SafeFrame implementations should either allow post messages or implement a proxy `__uspapi()` interface for a caller script within a sandbox that would otherwise be blocked. This proxy interface internally uses the SafeFrame messaging protocol to interface with the full implementation of the API on the publisher’s top frame and proxies responses back to the sandboxed caller. If allowing postMessage, vendors will not be required to accommodate any special protocols; they will simply use the postMessage method without SafeFrame.

If not allowing or blocking postMessage and therefore implementing the proxy method, vendors will see a local- to- the- sandboxed- iframe- scope `__uspapi()` proxy method that must behave the same as the asynchronous `__uspapi()` full-implementation method on the main publisher’s top frame.

**Without SafeFrames, using postMessage**

The `pwindow.postMessage()` method may be used from a child iframe to make requests send calls to from a parent’s (or any ancestor’s) frame’s API. To locate an ancestor frame capable of responding to postMessage() API calls search for an ancestor frame that has a child frame named '__uspapiLocator'.

**Sent Message**

The sent message should follow the form outlined below. The `command`, `parameter` and `version` object properties correspond to their namesake parameters defined as method
argument parameters for the __uspapi() method. The “sent message” also requires a unique callId property to help match the request with a response.

```json
{
    __uspapiCall:
    {
        command: "command",
        parameter: parameter,
        version: version,
        callId: uniqueId
    }
}
```

The returned message shall follow the form outlined below. The returnValue object property shall be the corresponding US Privacy String object for the command used upon sending the “sent message”. The success object property shall reflect the __uspapi() success callback argument and the callId will correspond to the “sent message” unique id passed in the callId property.

```json
{
    __uspapiReturn:
    {
        returnValue: returnValue,
        success: boolean,
        callId: uniqueId
    }
}
```