Global Privacy Platform

Streamlining technical privacy and data protection signaling standards into a singular schema and set of tools which can adapt to regulatory and commercial market demands across channels.

June 2022

Draft in Request for Public Comment until August 15th, 2022

This document has been developed by the Global Privacy Working Group.

IAB Tech Lab's Global Privacy Working Group will streamline technical privacy standards into a singular schema and set of tools which can adapt to regulatory and commercial market demands across channels. This group is open to members with technical expertise interested in contributing to development of industry privacy and data protection standards. This working group looks to the IAB Privacy Lab, local IABs and other relevant voices for policy requirements. This group will build on the foundations and experience of IAB Tech Lab's GDPR and CCPA Technical Working Groups. It will support the existing work and markets of those groups while expanding to cover new markets and channels.

Global Privacy Working Group Roster

At the date of publishing, Global Privacy Working Group Roster is made up of 376 individuals representing 163 organizations. Full roster details can be viewed here.

About IAB Tech Lab

Established in 2014, 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 solutions for brand safety and ad fraud; identity, data, and consumer privacy; ad experiences and measurement; and programmatic effectiveness. 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 Datalabel.org service. Board members/companies are listed at https://iabtechlab.com/about-the-iab-tech-lab/tech-lab-leadership/. For more information, please visit https://iabtechlab.com.

IAB Tech Lab Contact

Jason Raqueno
Sr. Director, Privacy & Data Protection

Table of Contents

1. Overview

About this RFC

IAB Tech Lab's Global Privacy Working Group is making considerable progress toward a Global Privacy Platform (GPP) architecture which will enable data use transparency and control for users, resulting in signals to the digital advertising supply chain supporting adherence to regional regulations and norms. This aim is clearly aligned with the public mission of the Global Privacy Working Group:

> Streamline technical privacy standards into a singular schema and set of tools which can adapt to regulatory and commercial market demands across channels.

Goals for RFC

- Formalize Tech Lab's intent to bring a GPP technical standard to market
- Solicit feedback on the architectural designs developed by the Global Privacy Working Group
- Use this feedback to catapult the Global Privacy Working Group forward in the directions most important to our industry and the broader public
- Demonstrate tangible progress toward supporting new markets like Canada and those covered by IAB's Cross Jurisdiction Privacy Project (CJPP), while also supporting existing markets like the EU (TCF) and US (USPrivacy), seeking to provide users with data use transparency and control
- Clearly link GPP to Tech Lab's simultaneous work on Addressability and Accountability in an industry experiencing major disruptions rooted in data protection and privacy concerns

Additional Reading and Reference Documents

- IAB Europe Transparency & Consent Framework Technical Specifications
- CCPA Compliance Framework Technical Specifications

GPP Value

Users

Internet users will see incremental predictability in transparency and control over time as privacy and data protection norms converge. Many more users, in countries previously uncovered by powerful digital advertising transparency and control tools, can see into and have a say over data uses for digital advertising.

Publishers, Advertisers and Ad Technology Products

Businesses will see reduced cost of maintaining privacy and data protection controls for users across the regions they work in. GPP adopters can adapt to regional changes and even gradual convergence in privacy and data protection experiences without technology switching costs.

2. Guiding Principles

- Reduce fragmented privacy signaling technologies that are costly for implementers
- Accommodate regional and country differences in privacy and data protection norms even if over time there is convergence
- Regions and countries should have support for their jurisdictions without needing to get other regions and countries to agree to adaptations
- Inclusive of existing in-use consent formats, recognizing that there have been a number of disparate efforts that need to be reflected
- The cost associated with adopting new privacy and data protection signaling technologies can be significant therefore we need to both minimizes costs and provide value with new features
- Flexibility is critical to support the needs of users, publishers, advertisers and the advertising technology products they use

3. Multi-Jurisdictional Design

There is a proliferation of data protection laws particular to countries and regions and we cannot ignore the issue of jurisdictional overlap of data protection laws. **These RFC designs acknowledge "jurisdictional overlap."** Jurisdictional overlap is a consequence of the extra-territorial nature of various data protection laws, including the GDPR which applies in the European Union but can also be applied in other jurisdictions under a few circumstances. In other words, data protection laws can be applied outside of the jurisdiction in which the law is adopted. For example, Country X's privacy law may apply to a user based in Europe who visits a country X digital property. This can result in a situation where an interaction falls within the scope of two or more jurisdictions, which could lead to participants, particularly digital properties and CMPs, needing to get consent compliant under multiple jurisdictions. Similarly, downstream vendors would need to consider which jurisdiction was relevant to their action. This can be problematic where there is a significant difference in requirements between different jurisdictions—for instance, some purposes may be specific to a jurisdiction, or there may be different legal requirements. Such situations could be supported by combining user preferences across different jurisdictions within a single signal.

To do this, policy inputs to date indicate that the technical architecture should clearly indicate which jurisdiction(s) are supported by a given GPP signal. The proposed approach, detailed below in the "Proposed Architecture", is to have sections (where sections equate to country or regional jurisdiction) within the GP string, where each section is dedicated to one jurisdiction with legal bases or permissions specific to that jurisdiction. CMPs only generate a GPP string

section for the jurisdiction(s) that the digital property requests and transmit that information to downstream vendors, along with an indication of the jurisdiction(s) that the digital property will apply. Participants thus make their own determination over how to proceed with the information provided to them via the digital property's CMP.

In practice, the string could support a number of sections. For example, if a user visits the same digital property, but from a new location in one of the supported GPP jurisdictions, that digital property can choose to apply the local jurisdiction's relevant policy framework and generate a new section in the GPP string specific to that jurisdiction. Vendors will have an indication that the digital property is now applying another jurisdiction to the transaction, and will also know the initial choice made in the original jurisdiction as well as the 'new' choice.

4. Proposed Architecture

"TC String" as a Starting Point

This proposal builds on the TCF v2.0 concept of a "TC String", composed of flexible and discrete "Segments", expanding these to support multiple existing and new privacy formats. This is our preferred path given the broad adoption of TCF v2.0 across digital properties, CMPs and ad technology products.

This architecture seeks to minimize disruptions to currently adopted privacy signaling, such as the TCF and USPrivacy, while at the same time giving potential GPP adopters reasons to make the update to their production implementations.

TC String Transport

As part of "TC String" as a starting point, the GPP string is also able to be transported via OpenRTB. While for TCF the jurisdiction information was housed in the Regs object and the string itself was housed in the User object, the GPP seeks to combine both the jurisdiction map and the user choice. This will begin as an extension within the Regs object:

Regs:

Attribute	Type	Description
ext.gpp	string	Contains the Global Privacy Platform's consent string. [Link to GPP doc]

Alternatives considered: User.ext.gpp; the user object seems a little limiting since some of the programs included may not be user-specific.

Basic Signal Makeup

GPP Sections encode:

- Disclosures to and control by the user, with a given granularity dependent on jurisdictional and policy requirements (by-purpose, by-vendor, by-legal-basis, or "omnibus" all or nothing)
- Metadata about the context of those choices (timestamps, versions, CMP info, locales, publisher info, regional or jurisdictional applicability, integrity signatures)
- Possible additional legal, publisher, or framework restrictions

Section specifications will clearly define which of the above data are represented, and in what form. Whenever possible, the various *technical* enumerations that have been developed for TCF v2.0 can be used directly or adapted:

- Integer identifiers for version & screens,
- CMP ID's from TCF v2.0
- Where applicable to the jurisdiction or region, the vendor ID's and data processing purposes as enumerated in the TCF v2.0 GVL can be reused in whole or adapted.

See “Discrete Sections” below for more detail.

Header

We are introducing a concept of a Header section on top of the TC String-inspired architecture. The purpose of the Header is to identify which regions’ transparency and control signals are included in a string payload and be a table of contents where to find each region in the string payload (broken into discrete sections). It is basically an ordered list of discrete sections that equate to different regions and counties and their jurisdictions. It lets readers understand what is present in the string and in what order. (See Discrete Sections below)

The header contains only a GPP version, the region ID(s) and index of the place of the associated region section in the string. The header delegates regional policy versions and technical encoding versions to each substring section so that each may develop independently of each other and the header design. (See Discrete Sections below)

Region IDs

This is an example of how Region IDs are enumerated. It is designed in a way to avoid needing to redesign anything about TCF upon which this proposal is based.

Section ID	Description
1	EU TCF v1 section (deprecated)

2	EU TCF v2 section (see note below)
3	GPP Header section (see note below)
4	GPP signal integrity section
5	Canadian TCF section
6	USPrivacy String *Unencoded Format* section
7	USPrivacy String *Encoded Format* segment. *See USPrivacy (Encoded Format) Example below.*
...	...

Note: *In order to make it simple to distinguish a GPP string from the existing IAB Europe TCF v2 TC String the first space in the header should be the version. This would allow current implementations to more easily understand and adapt to a GPP string. If the reader of a string finds "C" as the first character this indicates the string is IAB Europe's TCF v2.0 ("2" in bits corresponds to letter "C" in base64). If the reader of a string finds a "D" as the first character this indicates the string is GPP ("3" in bits corresponds to letter "D" in base64).*

Header Encoding

The Header consists of the following encoded fields:

Position	Type	Description
Type	Int(6)	Fixed to 3 as "GPP Header field"
Version	Int(6)	Version of the GPP spec (currently 1)
Sections	Range	List of Section IDs that are contained in the GPP string. Each ID represents a discrete Section that will be concatenated to the Header Section. The IDs must be represented in the order the related Sections appear in the string. This is required to make real time string processing less resource intensive.

Example GPP header for TCF EU (Range-Fibo encoding)

- Type: 3 (fixed) = 000011
- Version: 1 (fixed) = 000001

- Sections: 1x Range of [2]
  Range fields:
  - Amount: 000000000001
  - Item 1 Single: 0
  - Item 1 start ID: 011 (=2)

Full bit string:
000011 000001 000000000001 0 011

Encoded:

DBABMA

Full example including TC-String:

DBABMA~CPXxRfAPXxRfAAfKABENB-CgAAAAAAAAAAYgAAAAAAAA

Example GPP header for TCF EU + CCPA (Range- Fibo encoding)

- Type: 3 (fixed) = 000011
- Version: 1 (fixed) = 000001
- Sections: 2x Ranges of [2]+[6]
  Range fields:
  - Amount: 000000000010
  - Item 1 Single: 0
  - Item 1 start ID: 011 (=2)
  - Item 2 Single: 0
  - Item 2 offset to last ID: 1011 (= 4 => 4+2 = 6)

Full bit string:
000011 000001 000000000010 0 011 0 1011

Encoded:

DBACNYA

Full example including TC-String:

DBACNYA~CPXxRfAPXxRfAAfKABENB-CgAAAAAAAAAAYgAAAAAAAA~1YNN

Example GPP header for TCF CA + CCPA (Range- Fibo encoding)

- Type: 3 (fixed) = 000011
- Version: 1 (fixed) = 000001

- Sections: 1x Range of [5-6]
  Range fields:
  - Amount: 000000000001
  - Item 1 Single: 1
  - Item 1 start ID: 00011 (= 5)
  - Item 1 offset to last ID: 11 (= 1 => 5+1 = 6)

Full bit string: 000011 000001 000000000001 1 00011 11

Encoded:

```
DBABjw
```

Full example including TC-String:

```
DBABjw~1YNN~CPXxRfAPXxRfAAfKABENB-CgAAAAAAAAAAYgAAAAAAAA
```

Discrete Sections

IAB Europe's TCF v2.0 introduced the concept of Segments to privacy and data protection signaling. This design allows for flexibility needed by GPP. With certain tweaks described here, the TCF v2.0 TC String concept of discrete segments can be used to support multiple regions from one architecture while maintaining the ability to modify these discrete sections as needed.

Each string segment is scoped to the same body that updates the spec. This allows for regional sovereignty policies to make changes that might include more delimited information. For example, if TCF needs a version 3 and eliminates the concept of "out of band" vendors—which would result in the removal of DisclosedVendors and AllowedVendors—that should not require a version bump to the GPP string specification.

Delimiters

In order to be backward compatible with IAB Europe's TCF and USPrivacy string format, the delimiter used to separate segments is "~" (tilde).

> ***Note:*** *URL-safe characters are important to meet the integration needs of those not reading privacy signals server side or via the client side APIs. URL-safe characters are:*
>
> - *A-Z, a-z, 0-9*
> - *- (minus)*
> - *. (dot)*
> - *_ (underscore)*
> - *~ (tilde)*

"." and "-" and "_" are in use which leaves "~" as the only possible delimiter unless we re-use ".".

Section Encoding

Each region's discrete section is encoded according to that region's needs. This means today's implementations that read and adapt to TCF v2.0 signals or US Privacy signals don't need to change their logic for a given discrete section of the string, as long as the implementation is aware of where the discrete section is.

For new sections, the following guidelines are envisioned. Guidelines like these help developers quickly adopt new regions and allow for parsing new sections without the need to reinvent new data types. The format follows the encoding logic introduced by the TCF due to its prevalence in-market and because this RFC builds so heavily upon TCF v2.0 technical design.

Possible data types may include but are not limited to:

Data Type	Encoding	Description
Boolean	1 bit	0=true, 1=false
Integer (fixed length of x)	x bit	A fixed amount of bit representing an integer. Usual lengths are 3, 6 or 12 bit. Example: int(6) "000101" represents the number 5
Integer (Fibonacci)	Variable length	Integer encoded using Fibonacci encoding See "About Fibonacci Coding" below for more detail
String (fixed length of x) (including country codes)	x*6 bit	A fixed amount of bit representing a string. The character's ASCII integer ID is subtracted by 65 and encoded into an int(6). Example: int(6) "101010" represents integer 47 + 65 = char "h"
Datetime	36 bit	A datetime is encoded as a 36 bit integer representing the 1/10th seconds since January 01 1970 00:00:00 UTC. Example JavaScript representation: Math.round((new Date()).getTime()/100)

Bitfield (fixed length of x)	x bit	A fixed amount of bit. Usually each bit represents a boolean for an ID within a group where the first bit corresponds to true/false for ID 1, the second bit corresponds to true/false for ID 2 and so on.

Range (Int)	variable	A range field always consists of the following fields:
		1. int(12) - representing the amount of items to follow
		2. (per item) Boolean - representing whether the item is a single ID (0/false) or a group of IDs (1/true)
		3. (per item) int(16) - representing a) the single ID or b) the start ID in case of a group
		4. (per item + only if group) int(16) - representing the end ID of the group
		
		Example:
		int(12) = 2 // 2 items
		Bool = 0 // item 1 is type single ID
		int(16) = 3 // ID of item 1
		Bool = 1 // item 2 is type group
		int(16) = 5 // item 2 start ID
		int(16) = 8 // item 2 end ID
		
		Range = [3,5,6,7,8]
		Bits = 000000000010 0 0000000000000011 1 0000000000000101 0000000000001000
		
		Note: items may not be in sorted order.

Range (Fibonacci)	variable	A range field always consists of the following fields: 1. int(12) - representing the amount of items to follow 2. (per item) Boolean - representing whether the item is a single ID (0/false) or a group of IDs (1/true) 3. (per item) int(Fibonacci) - representing a) the single ID or b) the offset to the last end ID in case of a group 4. (per item + only if group) int(Fibonacci) - length of the group Example: int(12) = 2 // 2 items

		Bool = 0 // item 1 is type single ID
		int(Fibonacci) = 3 // ID of item 1
		Bool = 1 // item 2 is type group
		int(Fibonacci) = 2 // offset to last ID (3+2 = 5 is first ID)
		int(Fibonacci) = 3 // length of group (5+3 =>8 is last ID)
		
		Range = [3,5,6,7,8]
		Bits = 000000000010 0 0011 1 011 0011
		
		Note: items MUST be in sorted order.

When defining a new Section, regional policy writers should consider the above format in order to describe their segment.

Example

Example Field name	Example Type	Example Description
Version	int(6)	Version of Specification XYZ
LastUpdated	datetime	Datetime of last update
OptOutPurposes	Bitfield(6)	Purposes the user opted out for, each bit representing the purpose ID
...	...	...

Note: *It is recommended to use Field names in CamelCase and without any special chars or space. This allows the use of the same field names within other APIs (e.g. GPP JS API or GPP Mobile API)*

USPrivacy (Encoded Format) Example

In order to support backward compatibility, the section for USPrivacy is not encoded in the format below but in the original "1NYY" format (see "Region IDs" above). In order to align USPrivacy to the same encoding format, we propose a separate section definition for "USPrivacy (encoded format)" below.

Field name	Type	Description
Version	int(6)	The version of this string specification used to encode the string
Notice	Int(2)	Has notice been provided as required by 1798.115(d) of the CCPA and the opportunity to opt out of the sale of their data pursuant to 1798.120 and 1798.135 of the CCPA. Possible values: 0 = not applicable 1 = yes 2 = no
OptOutSale	Int(2)	Has user opted-out of the sale of his or her personal information pursuant to 1798.120 and 1798. If CPRACCPA applies, only 1 (yes) or 2 (no) can be used. Possible values: 0 = not applicable 1 = yes 2 = no
LSPACovered	Int(2)	Publisher is a signatory to the IAB Limited Service Provider Agreement(LSPA) and the publisher declares that the transaction is covered as a “Covered Opt Out Transaction” or a “Non Opt Out Transaction” as those terms are defined in the Agreement. Possible values: 0 = not applicable 1 = yes 2 = no

Sub-Sections / Segments

If a section uses sub-sections in order to separate information or to be more flexible, it can use the delimiter “.” (dot) to separate the sub-sections from each other. TCF v2.0 uses this method.

Data Use Taxonomy

Here we list all possible uses of data across all frameworks.

Categories

Category	Handle	Description
Purposes	purposes	Purpose means one of the defined purposes for processing of data, including users’ personal data, by participants in the Framework that are defined in the Policies or the Specifications for which Vendors declare a Legal Basis in the GVL and for which the user is

		given choice, i.e. to consent or to object depending on the Legal Basis for the processing, by a CMP
Special Purposes	specialPurposes	Special Purpose means one of the defined purposes for processing of data, including users' personal data, by participants in the Framework that are defined in the Policies or the Specifications for which Vendors declare a Legal Basis in the GVL and for which the user is not given choice by a CMP
Features	features	Feature means one of the features of processing personal data used by participants in the Framework that are defined in the Policies or the Specifications used in pursuit of one or several Purposes for which the user is not given choice separately to the choice afforded regarding the Purposes for which they are used
Special Features	specialFeatures	Special Feature means one of the features of processing personal data used by participants in the Framework that are defined in the Policies or the Specifications used in pursuit of one or several Purposes for which the user is given the choice to opt-in separately from the choice afforded regarding the Purposes which they support

Data uses

Category	Data Use	ID	Description
Purposes	Store and/or access information on a device	1	Cookies, device identifiers, or other information can be stored or accessed on your device for the purposes presented to you.
Purposes	Select basic ads	2	Ads can be shown to you based on the content you're viewing, the app you're using, your approximate location, or your device type
Purposes	Create a personalized ads profile	3	A profile can be built about you and your interests to show you personalized ads that are relevant to you
Purposes	Select personalized ads	4	Personalized ads can be shown to you based on a profile about you ads that are relevant to you
Purposes	Create a personalized content profile	5	A profile can be built about you and your interests to show you personalized content that is relevant to you

Category	Data Use	ID	Description
Purposes	Select personalized content	6	Personalized content can be shown to you based on a profile about you
Purposes	Measure ad performance	7	The performance and effectiveness of ads that you see or interact with can be measured
Purposes	Measure content performance	8	The performance and effectiveness of content that you see or interact with can be measured. be measured
Purposes	Apply market research to generate audience insights	9	Market research can be used to learn more about the audiences who visit sites/apps and view ads
Purposes	Develop and improve products	10	Your data can be used to improve existing systems and software, and to develop new products
Purposes	Sale of Personal Data	11	Sale of Personal Data according to CCPA

Possible states of data uses

Category	ID	Description
Consent	1	
Legitimate Interest	2	
Acknowledgement	3	
Opt Out	4	

Additional states can be defined based on needs, such as Acknowledgement and Opt Out.

Manifest

Given all the ingredients (the data uses) and how they can be combined (the possible states), the manifest is our recipe. Each manifest could be published as a json and/or on a simple web page.

Field	Description	Example TCF EU	Example TCF CA	Example USP
name		European Transparency and Consent Framework		
handle		tcfeu	tcfca	usprivacy
policyVersion	The current policy version, each framework should regulate what actions publishers must take in case it's bumped.	2	1	1
frameworkVersion	Each framework should regulate what actions publishers must take in case it's bumped.		1	1.1
gvlUrl		https://vendor-list.consensu.org/v2/vendor-list.json		
dataUses	Array of the data uses the framework allows	purposes {1,2,3,4,5,6,7,8,9,10}, specialPurposes {...} etc		purposes {11}
possibleStates		1, 2		3, 4

Supported Features		Restrictions, Lifetime disclosures		

Creating a GPP String

In order to create the GPP string:

1. For each section:
    0. When section is using the recommended base64-websafe encoding:
       Create a bit representation of the Section's header (if it exists) and each sub-section and convert them to base64-websafe without padding (i.e. removing "=" at the end) and concat the header and all sub-sections using "." (dot).
    1. When section is using a different encoding:
       Ensure that the data is websafe and does not include the "~" (tilde) character.
2. Create a bit representation of the GPP header section. Include all IDs for discrete sections in a sorted order. Convert it to base64-websafe without padding.
3. Concat the GPP-header as first item to the encoded versions of the discrete sections (step 2) using "~" (tilde) as delimiter.

Dealing with Length

TCF v2.0 implementers are finding many examples of strings too long for certain applications. An example case is when a digital property uses restrictions to set all vendors that allow it to consent, and the channel is via Accelerated Mobile Pages (AMP). A design that expects additional discrete Sections to be added to the TC String-inspired concept is likely to run into the same issues especially if certain regions' policies, and thus atomic section encodings, are very similar. This is already the case with IAB Canada's derivative of IAB Europe's TCF policy. To optimize this, we look at Fibonacci coding.

About Fibonacci Encoding

In simple terms, Fibonacci numbers are numbers that are the combination of the two Fibonacci numbers that come before. The numbers are [excluding 0, 1,] 1 (0+1), 2 (1+1), 3 (1+2), 5 (2+3), 8 (5+3), 13 (5+8), and so on... The numbers can be converted into bit sequences that are unique and will always end on "11". E.g. the bit representation of the first Fibonacci numbers are:

Number	Fibonacci representation	Bit sequence
1	F(2)	11
2	F(3)	011

3	F(4)	0011
4	F(2) + F(4)	1011
5	F(5)	00011
6	F(2) + F(5)	10011
7	F(3) + F(5)	01011

The advantage of this is, that a sequence of numbers can be encoded into a sequence of bits without the need to know the length of the bit sequences in advance. This, therefore, saves a lot of bits (size), in the case of TCF it especially saves size when encoding numbers that are around 100 to 4000.

Encoding and Decoding

The encoding and decoding is simple and can be done in basically all development languages.

(From Wikipedia)
To encode an integer *N*:

1. Find the largest Fibonacci number equal to or less than *N*; subtract this number from *N*, keeping track of the remainder.
2. If the number subtracted was the *i*-th Fibonacci number *F*(*i*), put a 1 in place *i*−2 in the code word (counting the left-most digit as place 0).
3. Repeat the previous steps, substituting the remainder for *N*, until a remainder of 0 is reached.
4. Place an additional 1 after the rightmost digit in the code word.

To decode a code word, remove the final "1", assign the remaining values 1,2,3,5,8,13... (the Fibonacci numbers) to the bits in the code word, and sum the values of the "1" bits.

GPP Consent Management APIs

Javascript

This API is meant to be a generic API that allows accessing consent information across legislations, regulations and standards. The intent is to have one “global” API that can be used to access the underlying “local” APIs such as IAB TCF and IAB CCPA/USP through a common interface. The goal is that new APIs can easily be defined by simply creating an API specification with a set of commands, without the need to define any additional javascript logic.

API prefixes

In order to distinguish between the different underlaying APIs, each API will be assigned a prefix. As of writing of this RFC, the following prefixes are defined:

API	Prefix
IAB TCF v1 (EU)	tcfeuv1 (no longer used)
IAB TCF v2 (EU)	tcfeuv2
IAB TCF v2 (Canada)	tcfcav2
IAB CCPA/USP v1	uspv1

API Interface

The API interface can be used to send/receive data to/from any underlying API. It is meant as a bridge across various APIs. The API interface is defined as follows:

```
__gpp(command, callback, parameter, [version])
```

Requirements to the interface:

- The function __gpp must always be a function and cannot be any other type, even if only temporarily on initialization – the API must be able to handle calls at all times.
- command must always be a string
- callback must be either a function or null
- parameter can be of mixed type depending on used command
- return value of the function is of mixed type depending on used command
- If a CMP cannot immediately respond to a query, the CMP must queue all calls to the function and execute them later. The CMP must execute the commands in the same order in which the function was called.
- A CMP must support all generic commands. All generic commands must always be available when a __gpp function is present on the page. This means, a so-called “stub code” that supports all generic commands must be in place before/during CMP load.

Generic commands

Generic commands are commands that can be used independent of underlying APIs/specifications. All generic commands must always be executed immediately without any asynchronous logic and call the supplied callback function immediately. The generic commands are:

ping

The ping command can be used in order to determine the state of the CMP. Example:

```
var PingReturn = __gpp('ping');
```

Argument	Type	Value
command	string	“ping”
callback	Not used	function(pingReturn: PingReturn, success)
parameter	Not used	

return	PingReturn object	

PingReturn

The PingReturn object is defined as follows:

```
PingReturn = {
gppVersion : String, // must be “Version.Subversion”, current: “1.0”
cmpStatus : String, // possible values: stub, loading, loaded, error
cmpDisplayStatus: String, // possible values: hidden, visible, disabled
apiSupport : Array of string, // list of supported APIs (prefix strings),
list may be empty during stub/loading. Example: ["tcfeuv2","uspv1"]
cmpId : Number, // IAB assigned CMP ID, may be 0 during stub/loading
}
```

addEventListener

The addEventListener command can be used in order to define a callback function that can be used to detect changes in the CMP. Example:

```
var EventListenerReturn = __gpp('addEventListener',myFunction);
```

Argument	Type	Value
command	string	“addEventListener”
callback	function	function (data: EventListener)
parameter	Not used	
return	EventListener object	

Note: The addEventListener command returns an EventListener object immediately. It will then call the callback function every time the CMP detects a change (see events below).

EventListener

The EventListener object is defined as follows:

```
EventListener = {
eventName : String, // possible values see events below
listenerId : Number, // Registered ID of the listener
data: mixed, // data supplied by the underlying API
}
```

Please note, that a call to the addEventListener command must always respond with a return object immediately. When registering new event listeners, the CMP (and stub) must therefore generate new listenerIds for each call to this command.

removeEventListener

The removeEventListener command can be used in order to remove an existing event listener. Example:

```
var EventListenerReturn = __gpp('removeEventListener',null, listenerId);
```

Argument	Type	Value
command	string	“removeEventListener”
callback	Not used	
parameter	Number	ID of the listenerId property that was returned from addEventListener command.
return	EventListener object	

Events

The following events may occur:

Event name	Type of data property	Description
listenerRegistered	Boolean	Only used within the return object to addEventListener command. The data property signals whether the event listener was registered successfully. If data equals true, a listenerId must be sent, otherwise the listenerId must be 0 (zero).
listenerRemoved	Boolean	Only used within the return object to removeEventListener command. The data property signals whether the event listener was successfully removed.
cmpStatus	String	Event is called whenever the status of the CMP changes (e.g. the CMP has finished loading). The data property will contain the new status (e.g. “loaded”)
cmpDisplayStatus	String	Event is called whenever the display status of the CMP changes (e.g. the CMP shows the consent layer). The data property will contain the new display status (e.g. “visible”)
error	String	Event can be called by the CMP in case of an error. The data property will contain a human readable error message.

[API-prefix] or [API-prefix].[Eventname]	mixed	Event is called by the CMP depending on the specific API needs (e.g. IAB TCF EU may specify different events than IAB USP). If the API defines different event types, these can be used by combining API-prefix and eventname. If the API does not specify different event types (e.g. in IAB TCF v2.0), the API-prefix is used as name. The data property will contain mixed data depending on API.

hasSection

The `hasSection` command can be used in order to detect if the CMP has generated a section of a certain specification. For example a client can ask the CMP if data for IAB TCF v2.0 is existing or not. Example:

```
var b = __gpp('hasSection',null, "tcfeuv2");
```

Argument	Type	Value
command	string	“hasSection”
callback	Not used	
parameter	String	API Prefix string
return	Boolean	True if the specified section is present, otherwise false.

getSection

The `getSection` command can be used in order to receive the (parsed) object representation of a section of a certain specification. For example a client can ask the CMP to get the IAB TCF v2.0 TCData. Example:

```
var s = __gpp('getSection',null, "tcfeuv2");
```

Argument	Type	Value
command	string	“getSection”
callback	Not used	
parameter	String	API Prefix string
return	Object	Section string parsed into an object according to API specification.

getField

The `getField` command can be used in order to receive a specific field out of a certain section. For example a client can ask the CMP to get the last updated field from the IAB TCF v2.0 TCData. Example:

```
var s = __gpp('getField',null, "tcfeuv2.LastUpdated");
```

Argument	Type	Value
command	string	“getField”
callback	Not used	
parameter	String	API Prefix string + “.” (dot) + fieldname
return	mixed	Depending on API specification.

Non-generic commands

Besides the generic commands, a CMP can support other commands that are defined by any underlying API (e.g. IAB TCF EU, IAB TCF Canada, IAB USP/CCPA, …). Any API that wants to make use of the GPP API spec should define a set of commands as follows:

Example command xyz

Command: **prefix.command**
Callback: **function (…)** **or** **“Not used”**
Parameter: **data type** **or** **“Not used”**
A description to the command, what does it do, what is it meant for, when to use it and how.

For example the IAB TCF v2.0 getTCData command would be defined as:

getTCData

Command: **iabtcfeuv2.getTCData**
Callback: **function(tcData: TCData, success: boolean)**
Parameter: **(optional) int array**
The vendorIds array contains the integer-value Vendor IDs for Vendors in which transparency and consent is being requested. …

In the above example a call to `__gpp('iabtcfeuv2.getTCData',myfunction)` will be translated by the CMP to `__tcfapi('getTCData',2,myfunction)`.

Usage example

The following table demonstrates two code versions for checking consent with the IAB TCF v2.0 API (differences in red):

Use of IAB TCF v2.0 API	Use of GPP API

`<script>`
`  if (window.__tcfapi)`
`  {`
	`<script>`
	`  if (window.__gpp)`
	`  {`

```
    // add IAB TCF event listener
    __tcfapi('addEventListener', 2,
    function (tcData, success)
    {
      if (success &&
          tcData.eventStatus ===
'tcloaded')
      {
        // do something with
tcData.tcString

        // remove the ourself to not get
        // called more than once
        __tcfapi('removeEventListener', 2,
        function (success)
        {
          if (success)
          {
            // oh good...
          }
        }, tcData.listenerId);
      }
      else
      {
        // do something else
      }
    });
  }
</script>
```

```
    // add IAB TCF event listener
    __gpp('tcfeuv2.addEventListener',
    function (tcData, success)
    {
      if (success &&
          tcData.eventStatus === 'tcloaded')
      {
        // do something with tcData.tcString

        // remove the ourself to not get
        // called more than once
        __gpp('tcfeuv2.removeEventListener',
        function (success)
        {
          if (success)
          {
            // oh good...
          }
        }, tcData.listenerId);
      }
      else
      {
        // do something else
      }
    });
  }
</script>
```

```
<script>
  if (window.__gpp)
  {
    // add generic event listener
    __gpp('addEventListener', function (ev)
    {
      if (ev.eventName == 'tcfeuv2')
      {
        var tcData = ev.data; //IAB TCF event
data
        if (tcData.eventStatus === 'tcloaded')
        {
          // do something with tcData.tcString

          // remove the ourself to not get
          // called more than once
          var r = __gpp('removeEventListener',
                        null, ev.listenerId);
          if (r.data == true)
          {
            //event handler was removed
          }
          else
          {
            //issue removing event handler
          }
        }
        else
        {
          // do something else
        }
      }
      else
      {
        // event is outside of the IAB TCF v2,
        // maybe CCPA or generic ...
      }
    });
  }
</script>
```

Non-web (Mobile, CTV, etc)

How is GPP used in-app?

Similar to the support for TCF, for GPP we will standardize storage locations and naming for the content of the GPP data and GPP string so that ad tags embedded in mobile apps can find the GPP data and string in a consistent way.

The pre-parsed GPP data as well as the GPP string shall be stored under NSUserDefaults(iOS) or SharedPreferences(Android). This will allow:

- Vendors to easily access GPP data
- GPP data to persist across app sessions

- GPP data to be portable between CMPs to provide flexibility for a publisher to exchange one CMP SDK for another
- Vendors within an app to avoid code duplication by not being required to include a GPP string decoder while still enabling all typical use cases
-

Note: *If a Publisher chooses to remove a CMP SDK from their app they are responsible for clearing all IABGPP_* vestigial values for users so that vendors do not continue to use the GPP data therein.*

The key names will be a combination for the IABGPP_ prefix followed by the section prefix followed by an underline and then followed by the name of the value it represents. The relevant sections documented in this spec are HDR (header), TCFEU (tcfv2), TCFCA (Canadian TCF), USPRIVACY (U.S. Privacy).

There will be additional keys added to the IABGPP_HDR for things like CMP SDK ID, or CMP SDK Version.

The data stored in the in-app storage is not encoded or compressed because the storage is private to the application. The resulting string however will follow the encoding rules as listed in this spec.

Example of key names:

The GPP version stored in the header will be named IABGPP_HDR_Version. The full compress string will be IABGPP_GppString. The U.S. Privacy string will be stored under the key name GPP_USPrivacy_String, for instance the TCF EU vendor consent would be IABGPP_TCFEU_PurposeConsents, etc. A full list of all the names will be published in the final specification.

How do third-party SDKs (vendors) access the consent information in-app?

On both Android OS and iOS, the vendor can get notified when the values of the shared keys change. See NSUserDefaultsDidChangeNotification and SharedPreferences.OnSharedPreferenceChangeListener.

On Android OS, the GPP data and GPP string shall be stored in the default Shared Preferences for the application context. This can be accessed using the getDefaultSharedPreferences method from the android.preference.PreferenceManager class using the application context. The GPP data values can be retrieved from the application Shared Preferences by key name using the get methods on the android.content.SharedPreferences class.

Signal Integrity

As part of the first version of GPP, signal integrity will be accomplished in concert with the Accountability Platform. The Global Privacy Working Group is committed to introducing signal integrity technology for GPP in future versions. Right now, the group is evaluating the following proposals for signal integrity.

- Cryptographic Security Foundations for Programmatic Ads Ecosystem
- JWT Consent Token Proposal
- Proposals for TCF Signal Integrity that could be applicable to GPP

GPP Identifier

We recognize there are existing vendor lists (see note with non-exhaustive list below) on which the same business entity may appear. In the near-term it is recommended that close cousins to TCF (such as IAB Canada's TCF policy) build upon the current Global Vendor List (GVL). Note that the proposed GPP string architecture is supported by the GVL and CMP list. This approach enables the use of the current Global Vendor ID, for all vendors who have registered, and the creation of con-current non-overlapping vendor IDs for new vendors, specifically for TCF-like/derived policy - in the short-term this will simplify the adoption of the GPP. TCF-like or derived policy is described as a direct or near direct adoption of TCF's purposes, special purposes, features and special features.

In these instances the registration would be governed by the local jurisdiction Policy and T&Cs, who would also be responsible for enforcement and compliance. Vendors looking to register should do so on the registration portal.

For non TCF-like approaches, Tech Lab's Transparency Center, which is already hosting the Limited Service Provider Agreement (LSPA) signatory list, could be a central API that pulls from local privacy and data protection signaling policy registries.

In future versions of GPP, the central API could also include a business entity identifier generator. This would allow callers needing to consume privacy signals with business entity level disclosures across multiple markets the ability to do so with the assurance that business entities do not have duplicate IDs (or overlapping).

The Registry API

Registry Directory

This endpoint allows those who need to query for information about business entities across regions a central place to see a comprehensive view of that information. This can either be in the form of a sort of phone directory, directing callers to the right local APIs or as an aggregator of information from the local APIs.

Known vendor lists:

- TCF v2.0 GVL
- Google Additional Consent (companies on Google's ATP list not part of TCF v2.0)
- NAI opt out
- DAA opt out
- Limited Service Provider Agreement Signatory List (includes publishers and advertisers)

Unified Libraries and SDKs

It is our intention to support official open-source projects to aid integration to this spec. Existing official open-source projects like iabtcf-es could be extended and consolidated to work with GPP.

5. Vision for Updates from TCF v2.0 and/or USPrivacy to GPP

We reiterate the intention of this proposed architecture is to support privacy and data protection signaling world-wide. That means the proposed architecture supports TCF v2.0 and USPrivacy in addition to new markets. The future finalization of any GPP technical specification does not equate to an end of life for region-specific technical specifications outside of GPP. However, it is the Global Privacy Working Group's aim to design GPP in such a way that transition would be straightforward and as low cost as possible. It is also the working group's aim for GPP to provide value that leads market participants to adopt it. This way, after a period of time, we are left with only a single technical standard to maintain and support.

6. How to Submit Comments

Comments on this RFC may be submitted to globalprivacy@iabtechlab.com.

7. Appendix

European TCF with Canadian TCF Signal Definition