TrustArc

Cookie Consent Manager

API Integration Guide

Revision History

Document Version	Date	Description
1.0	02-16-2024	First revision of the rebranded document.
1.1	02-19-2024	Added important note for page refresh
1.2	03-24-2024	Added Verification and Troubleshooting
1.3	08-08-2024	Updated the Consent Manager API Functions section
1.4	11-11-2024	Added getDefaultCategories API
1.5	05-06-2026	Added Data Layer Event section and removed references to &gtm=1

About This Document

This document provides access to Consent Manager methods that a customer can use to gain the consent level and other useful information from a user's interaction with TrustArc's Consent Manager.

Target Audience

This document is intended for implementation managers and/or web development teams.

Disclaimer

Information in this document is subject to change without notice. No part of this document may be reproduced or transmitted in any form by any means, electronic or mechanical, for any purpose, without the express written permission of TrustArc Inc. Moreover, this guide is strictly informational in nature and **is not intended to provide legal advice**, as all legal and compliance obligations and interpretations remain the responsibility of users in coordination with their legal counsel.

Table of Contents

Consent Manager API Functions

Companies may first use TrustArc's Website Monitoring Service to crawl their websites to capture all tracking technologies and the originating HTTP requests in order to help identify tags and gateway elements. Based on the level of user consent received via Cookie Consent API, companies are able to manage tags on their websites.

Below is a list of the API functions that are available in the Consent Manager that may provide the functionality necessary to help customers utilize the user's consent choices in a customized and unique manner fitting for their website and needs to provide users with their consent choices.

getConsent(...)

Description	This method provides the ability to ascertain the opt-out status for a specific vendor domain. **Note**: This is for the use of the website hosting Consent Manager and requesting specific vendor opt outs.		
callApi Parameters	Name	Value	Description
	`getConsent`	(string)	[required]
	`self`	(string)	[required] This represents the API caller (current website).
	`domain`	(string)	[optional] The vendor domain you are requesting opt in status for Default Value, which if left blank, the value is assumed to be the same as *self.location.hostname*
	`authority`	(string)	[optional]

			The domain name of the source from which the API caller derives permission to make the call
	`type`	(string)	[optional] The bucket type you are requesting opt in status for; values can be: • default types - "required," "functional," or "advertising" • bucket/category name - e.g. "Required Cookies" **NOTE:** This can only be used if there is already a consent. • bucket/category number - index starts at 1, e.g. "1" for Required Cookies
Sample JS Function Request			`var json = truste.cma.callApi("getConsent", "yourdomain.com", "targetVendorDomain.com", "authorizingBodyDomain.com", "functional");` **NOTE:** If both `targetVendorDomain.com` and `functional` are specified and have conflicting opt-in status, the result will be the consent provided for the specified bucket (`functional`).
Sample Response			`json = {source:"asserted", consent:"denied"};` Where: • `source` will have the following values: ○ **asserted** – The consent is expressed - i.e. achieved by an explicit user action (such as a mouse click) in an implied or expressed consent notice. ○ **implied** – The consent is implied - i.e. achieved via displaying a corresponding notice to the user where a user has not taken any explicit action, or set as a default preference by the

	website owner.
	• `consent` will have the following values:
	  ○ **approved** – based on the user preference, the party is allowed to set cookies and other related technologies
	  ○ **denied** – based on the user preference, the party is NOT allowed to drop cookies and other related technologies
	Default Values (prior to consent):
	• If the Consent Manager's behavior is expressed:
	  ○ `{source:"implied", consent:"denied"};`
	• If the Consent Manager's behavior is implied:
	  ○ `{source:"implied", consent:"approved"};`

getConsentDecision(...)

Description	This method provides the ability to ascertain the consent level of the user that provided consent on your website. This function only returns the highest category index that the user opted in.		
callApi Parameters	Name	Value	Description
	`getConsentDecision`	(string)	[required]
	`self`	(string)	[required] This represents the API caller (current website).
Sample JS Function Request	`var json = truste.cma.callApi("getConsentDecision", "yourdomain.com");`		

Sample Response

```
json = {consentDecision:$integer, source:"asserted"};
```

Where:

- `consentDecision` values:
  - `$integer` = index of the highest category opted in; category index starts with 1, that is, by default:
    - `1` = required
    - `2` = functional
    - `3` = advertising
  - `0` = no preference set

  NOTE: The value returned is the highest category opted in. In the case where `functional` is opted out but `advertising` is opted in, the `consentDecision` value will still be 3.

- `source` will have the following values:
  - asserted – The consent is expressed - i.e. achieved by an explicit user action (such as a mouse click) in an implied or expressed consent notice.
  - implied – The consent is implied - i.e. achieved via displaying a corresponding notice to the user where a user has not taken any explicit action, or set as a default preference by the website owner.

  NOTE: If you have custom bucketing, you should factor in the number associated with your customized naming convention that should be returned in the `$integer` value.

  Please see the *Appendix* section for the possible use case and resulting values this function can provide.

getGDPRConsentDecision(...)

<table>
<tr><td>Description</td><td colspan="3">This method provides the ability to ascertain the consent level that the user chose to a granular level. This method takes into account the multiple preference values that a user can now choose.</td></tr>
<tr><td rowspan="3">callApi Parameters</td><td>Name</td><td>Value</td><td>Description</td></tr>
<tr><td><code>getGDPRConsentDecision</code></td><td>(string)</td><td>[required]</td></tr>
<tr><td><code>self</code></td><td>(string)</td><td>[required]
This represents the API caller (current website).</td></tr>
<tr><td>Sample JS Function Request</td><td colspan="3"><code>var json = truste.cma.callApi("getGDPRConsentDecision","yourdomain.com");</code></td></tr>
<tr><td>Sample Response</td><td colspan="3"><code>json = {consentDecision: $integer_array, source: "asserted"}</code>

Where:
• <code>consentDecision</code> values:
○ <code>$integer_array</code> = provides the granular consent levels (represented by category indices) that the user selected. For example, if the user opted in to required and advertising, but opted out of functional, the value will be <code>[1,3]</code>
○ <code>[0]</code> = no preference set
• <code>source</code> will have the following values:
○ asserted – The consent is expressed - i.e. achieved by an explicit user action (such as a mouse click) in an implied or expressed consent notice.
○ implied – The consent is implied - i.e. achieved via displaying a corresponding notice to the user where a user has not taken any explicit action, or set as a default preference by the</td></tr>
</table>

	website owner. Please see the *Appendix* section for the possible use case and resulting values this function can provide.

getConsentCategories(...)

Description	This method provides the ability to retrieve all the preference values set by an end user for all vendors.		
callApi Parameters	Name	Value	Description
	`getConsentCategories`	(string)	[required]
	`self`	(string)	[required] This represents the API caller (current website).
Sample JS Function Request	`var json = truste.cma.callApi("getConsentCategories","yourdomain.com");`		

Sample Response	Each domain/vendor will have one of the following values:
	• `0` – means opted-out
	• `1` – means opted-in
	• `2` – means no preference
	
	`json =`
	`{`
	`categories:{`
	`"Required Cookies":{`
	`"value":"0",`
	`"domains":{`
	`"bluekai.com":"2",`
	`"partner.bluekai.com":"2"`
	`}`
	`},`
	`"Functional Cookies":{`
	`"value":"1",`
	`"domains":{`
	`"2o7.net":"1",`
	`"everesttech.net":"1"`
	`}`		

```
        },
        "Advertising Cookies":{
            "value":"2",
            "domains":{
                "ads.creative-serving.com":"0",
                "yieldoptimizer.com":"0",
                "adaptv.advertising.com":"0",
                "adap.tv":"1"
            }
        }
    }
}
```


NOTE: If there is no consent yet, the object returned will be `{categories: 'no categories'}`

getDefaultCategories

Description	This provides the ability to fetch the default category settings configured for a specific country, even before a user provides consent.		
callApi Parameter	Name	Value	Description
	`getDefaultCategories`	(string)	[required]
Sample JS Function Request	`truste.cma.callApi('getDefaultCategories','<cminstanceid>');`		

Sample Response	`country: "us"`
	`defaultCategory: "Functional Cookies"`
	`state: "az"`		

setConsentLevel

Description	This method provides the ability to set a consent programmatically for the end user. When consent is set using this API, the consent manager banner/modal will be dismissed.		
callApi Parameters	Name	Value	Description
	`setConsentLevels`	(string)	[required]
	`yourdomain.com`	(string)	[required] This represents the API caller (current website).
	`consentArray`	[array]	[required] represents an array of consent category levels that will be opted-in.
Sample JS Function Request	`var json = truste.cma.callApi('setConsentLevels', 'truste.com ', [1,2,3]);`		
Sample Response	Each element in the array corresponds to a consent category set in the Consent Manager. For the default category, 1 represents Required Cookies, 2 represents Functional Cookies, and 3 represents Advertising Cookies. **NOTE:** Please check the Appendix for sample use cases.		

Data Layer Event

The Data Layer Event feature allows the Consent Manager to push event data to the `dataLayer`. This enables third-party tools, such as tag management systems, to detect and respond to a user's consent state and any updates to their preferences.

Prerequisite

Ensure that the **Push CCM events to dataLayer** option is enabled in your Consent Manager settings.

How It Works

When this feature is enabled, the Consent Manager automatically sends event data to the `dataLayer`in the following scenarios:

- When the Consent Manager loads and initializes
- When a user updates their consent preferences

Event Payload Structure

The payload for these events adheres to the following JSON structure:

```
JSON
{
       'event': 'trustarc-ccm-ready' | 'trustarc-consent-updated',
       'timestamp': <timestamp>,
        'consent': {
          ta_category_1: true | false | undefined,
          ta_category_2: true | false | undefined,
          ta_category_3: true | false | undefined,
          ta_category_n: true | false | undefined,
        },
       'consentModel': 'opt-in' | 'opt-out' | 'none'
}
```

Event Types

trustarc-ccm-ready

This event is triggered when the Consent Manager has fully loaded on the page. It provides the user's initial or existing consent state.

trustarc-consent-updated

This event is triggered whenever a user updates their consent preferences. This includes:

- When a user clicks the **Agree and Proceed** button.
- When a user finishes adjusting their preferences by clicking **Submit Preferences** after going to the Overlay screen.
- When a user changes previous settings by clicking **Submit Preferences** after manually invoking the Consent Manager.

Payload Fields

The following are the payload fields:

- event - The name of the event dispatched to the `dataLayer`.
- timestamp - The time (in milliseconds) when the event was recorded.
- consent - An object containing the user's consent decisions for each configured cookie category.
    - `true` — Consent has been granted (opt-in)
    - `false` — Consent has been denied (opt-out)
    - `undefined` — No preference has been set
- ta_category_n - Represents a specific cookie category index
    - For example: `ta_category_1` for Required Cookies
- consentModel - Indicates the consent model applied to the user's session.

- `opt-in` — Consent must be explicitly granted before cookies are set
- `opt-out` — Consent is assumed unless explicitly denied
- `none` — No consent model is applied

NOTE: The dataLayer events can be used to trigger tag execution or suppression based on user consent. Additionally, ensure that your tag management system is configured to listen for these events.

Event Handlers to Retrieve Notifications from Consent Manager

TrustArc provides the capability to notify a third-party on the publisher's site, publisher itself, or the tag management system of certain events that have occurred with the Consent Manager tool. If you choose to be notified of certain events such as receiving the User's consent level, looking up for a specific vendor opt out, and/or trying to get information from within a Third Party IFrame, please see below for the possible methods to use.

Method to Retrieve User Level Consent

To receive this notification, pass a Javascript callback function in the following way:

```
None
var apiObject = {PrivacyManagerAPI:
      { action:"getConsent",
        timestamp: new Date().getTime(), self: "domainThatIsAsking.com" }};

var json = JSON.stringify(apiObject); window.top.postMessage(json,"*");
window.addEventListener("message",  yourMessageHandler,  false);
```

Where `yourMessageHandler` is the javascript function that handles PostMessage events. Please see the sample section below regarding implementing this method to see an example of the response.

TrustArc Cookie Consent Manager will dispatch events in the following cases:

1. When a user clicks on **Agree and Proceed** button.
2. When a user finishes adjusting his/her preferences by clicking **Submit Preferences** after going to *Overlay* screen.
3. When a user changes the previously set (or implied) settings by clicking **Submit Preferences** after manually invoking TrustArc Cookie Consent Manager from the publisher's site.
4. When an implied preference is set outside of the Consent Manager modal window, for example, the banner.

Using TrustArc API from Third-Party iFrame

For security purposes, not all third parties on the sites where publishers integrated TrustArc Cookie Consent Manager will be allowed to query user preferences about domains other than their own from inside third-party iFrames. This is because user privacy preferences can be considered private information per se. They are intended only for the companies they apply to or for the companies who are capable to act upon them to enforce user preferences such as tag management companies. Therefore, all callers who wish to request consent for domains other than their own (i.e. other than the "origin" attribute of the PostMessage event) are required to obtain prior authorization from an authorizing body such as TrustArc to call this API in an asynchronous way.

Important: TrustArc will log all calls to its API on its server and will blacklist the iFrame domain names of the third parties whom it believes have not obtained the proper authorization to use this API.

Below is a typical example of how to make a previously mentioned `getConsent` call from a third-party iFrame.

```
None
var apiObject ={PrivacyManagerAPI:
       {action:  "getConsent",
       timestamp: new Date().getTime(),
       domain:  "targetDomain.com"
       self:   "domainThatIsAsking.com",
       authority:   "authorizingBody.com",
       type: "type1, type2, type3, ..."}};
var json = JSON.stringify(apiObject); window.top.postMessage(json,"*");
window.addEventListener("message",handleMessage,false);
```

This method accepts exactly the same parameters as described above for the synchronous `getConsent` call. However, when a call is made from inside an iFrame, TrustArc will assume the caller's identity based on the iFrame properties automatically set by the browser (i.e. the "origin" property of the PostMessage). If the request is for the user preferences for the same domain, the `authority` parameter will be ignored.

How to Implement HandleMessage Function

Below is an example of how callers could implement `handleMessage` function:

```
None
handleMessage(e){
	var json = JSON.parse(e.data);
	if(json &&
	json.PrivacyManagerAPI){
		switch(  json.PrivacyManagerAPI.consent){
			case "denied":
				break;
			case "approved":
				break;
		}
	}
}
```

The returned json will have the following properties:

```
None
{PrivacyManagerAPI:
	{capabilities:["getConsent"],
	source:"asserted",
	consent:"denied",
	action:  "getConsent",
	timestamp: 1234567890,
	domain:  "targetDomain.com"
	self:   "domainThatIsAsking.com",
	authority:   "authorizingBody.com",
	type: "type1, type2, type3, ..."}};
```

Where:

- `source` and `consent` properties will have the same range of values as described above
- `action`, `domain`, `self`, `authority`, and `type` properties will simply duplicate the input parameters in the initial asynchronous method calls
- `capabilities` will be an array of call names TrustArc has authorized the caller to make. (Right now this is simply a list of all calls supported by the API. This field can also be used for versioning purposes. TrustArc may change it in future versions of this API).

Note: To prevent any unnecessary delays, TrustArc will send the response immediately as soon as it knows user preference - not after it would log the call and verify whether the caller is authorized to make it. If it finds later that the caller did not obtain proper authorization, it will blacklist the caller and not respond to all future calls from this company.

Zero Cookie Load Integration Using Consent Manager API

A Zero Cookie/Tracker load integration with Consent Manager indicates that you wish to hold back all firing of any tags that do not meet or belong in the Required Cookies Bucket within your Consent Manager instance. For example, a user visiting your site for the first time will only be served content and code that will drop ONLY required or strictly necessary cookies for the website to function properly.

Normally a Tag Management System (a system that can be used to integrate third-party software) is utilized to help achieve a Zero Cookie Load integration, but other technologies can be used as well. We do not recommend or restrict our clients from using any other technologies to achieve this functionality.

The CM API can be used to ascertain the consent level of the user and based on that consent level, whether it be a bucket level consent or more granular, you can utilize this consent preference and pass that back accordingly to your technology of choice to ensure that Zero Cookie Load is being respected. The API will be the mechanism to grab the user consent, vendor opt-outs, or specific events you will need to listen for in order to properly signal your technology to respect the choice of the user and not fire those third-party tags and other code.

Verification and Troubleshooting

This section outlines how to verify a Zero-Tracker integration and some common troubleshooting steps.

Verifying a Zero-Tracker Integration

The process of verifying your CM API Integration will vary depending on the specifics of your deployment, CM configuration, and overall tools at your disposal. However, in general you will be testing that the correct blocking occurs in three situations:

- A new user with Standard Load
- A new user with Zero-Tracker Load
- Changing the existing consent for each category

A new user with Standard Load

When verifying a new user who should see a Standard Load, we are simply ensuring that all tags are fired as expected on the initial load. You should see the same tags fire after the integration as before the integration.

A new user with Zero-Tracker Load

When verifying a new user who should see a Zero-Tracker Load, we are verifying that all of the tags that have the blocking conditions applied have NOT been fired when coming from a region requiring a Zero-Tracker Load. This should be done before any user interaction with the Consent Manager.

Once consent has been submitted, and if consent has been provided, the appropriate tags should fire on the next page load.

Changing the existing consent for each category

When verifying for a user who has changed their consent we are verifying that the correct tags are blocked or fired based on the change. For example, when verifying the change in consent from Advertising to Functional we would expect to see any Advertising tags NOT to be fired on the next page load. Conversely, a change from Functional to Advertising should cause any Advertising tags to be fired. A change both to and from each consent category is recommended.

Important: To ensure consent preferences are appropriately honored, once a tag script or code has been executed on a page (i.e., a tag has fired), you will need to configure an automatic page refresh in order for the new consent preference to be reflected. Failure to execute a refresh will result in the prior tracking behavior (e.g., tracking will continue as if an opt-out had not occurred) to persist until a manual refresh is done by a web visitor themselves.

CM API Demonstration

We have built a demo website where the CM API is being used to control third party tags so that these trackers will fire as per the consent preference of the user.

Appendix

This section provides the common use case scenarios that a user may provide consent through their interaction with the TrustArc Consent Manager. Through these possible use cases, we also detail the possible results returned by the consent manager API calls. When referencing the table result values, please reference the use case scenario below for more details on when or how this might be collected.

Use Case Scenarios

Description	Opt-In User Action
1	When an EU user opens the page having TrustArc Consent Manager for the first time, the banner is added to the footer.
2	User closes the banner.[1]
3	User selects the **Required Cookies** level on TrustArc Consent Manager dialog using the *Slider/Radio* button.
4	User selects the **Functional Cookies** level on TrustArc Consent Manager dialog using the *Slider/Radio* button.
5	User selects the default option by clicking **Agree and Proceed** on TrustArc Consent Manager dialog. OR User selects the **Advertising Cookies** level on TrustArc Consent Manager dialog
6	User selects the **Required Cookies** level on TrustArc Consent Manager Advanced Settings or Granular Consent UI.

[1] **NOTE:** For scenario 2, it is considering that the **Close** button is the same as the **Decline All**. Implementations for the **Close** button may change based on the customer's settings.

7	User selects the **Required, and Functional Cookies** level on TrustArc Consent Manager Advanced Settings or Granular Consent UI.
8	User selects the **Required, Functional, and Advertising Cookies** level on TrustArc Consent Manager Advanced Settings or Granular Consent UI.
9	User selects the **Required, and Advertising Cookies** level on TrustArc Consent Manager Advanced Settings or Granular Consent UI.

getConsentDecision Results Table

If you are using the Slider version of the Consent Manager UI, these are the potential results that you will receive from the APIs:

Use Case	Source Value	CMAPI Value	Message from TrustArc	cmapi_cookie_privacy	cmapi_gtm_bl
1	implied	0	{consentDecision:[0], source:"implied"};	<empty>	<empty>
2	asserted	1	{consentDecision:[1], source:"asserted"};	permit 1 required	ga-ms-ua-ta-asp-bzi-sp-awct-cts-csm-img-flc-fls-mpm-mpr-m6d-tc-tdc
3	asserted	1	{consentDecision:[1], source:"asserted"};	permit 1 required	ga-ms-ua-ta-asp-bzi-sp-awct-cts-csm-img-flc-fls-mpm-mpr-m6d-tc-tdc
4	asserted	1,2	{consentDecision:[2], source:"asserted"};	permit 1,2 functional	ta-asp-bzi-sp-awct-cts-csm-img-flc-fls-mpm-mpr-m6d-tc-tdc
5	asserted	1,2,3	{consentDecision:3, source:"asserted"};	permit 1,2,3	<empty>

6	asserted			Same as Use Case #3
7	asserted			Same as Use Case #4
8	asserted			Same as Use Case #5
9	asserted			Same as Use Case #5

getGDPRConsentDecision Results Table

If you are using the Granular version of the Consent Manager UI, these are the potential results that you will receive from the APIs:

Use Case	notice_gdpr_prefs cookie value	CMAPI Value	Message from TrustArc	cmapi_cookie_privacy	cmapi_gtm_bl
1	<empty>	0	{consentDecision:[0], source:"implied"};	<empty>	<empty>
2	-1	0	{consentDecision:[0], source:"asserted"};	<empty>	<empty>
3	0	1	{consentDecision:[1], source:"asserted"};	permit 1 required	ga-ms-ua-ta-asp-bzi-sp-awct-cts-csm-img-flc-fls-mpm-mpr-m6d-tc-tdc
4	0,1	1,2	{consentDecision:[1,2], source:"asserted"};	permit 1,2 functional	ta-asp-bzi-sp-awct-cts-csm-img-flc-fls-mpm-mpr-m6d-tc-tdc
5	0,1,2	1,2,3	{consentDecision:[1,2,3], source:"asserted"};	permit 1,2,3	<empty>

6	0	1	{consentDecision:[1], source:"asserted"};	Same as Use Case #3
7	0,1	1,2	{consentDecision:[1,2], source:"asserted"};	Same as Use Case #4
8	0,1,2	1,2,3	{consentDecision:[1,2,3], source:"asserted"};	Same as Use Case #5
9	0,2	1,3	{consentDecision:[1,3], source:"asserted"};	Same as Use Case #5

setConsentLevels Results Table

If you are using this API to programmatically set consent for the user, below are the results in different use cases.

Sample Value	Expected Results (notice_gdpr_prefs)	Expected Results (cmapi_cookie_privacy)
truste.cma.callApi('setConsentLevels', 'truste.com ', [1,2,3]);	0,1,2	permit 1,2,3
truste.cma.callApi('setConsentLevels', 'truste.com ', [1]);	0	permit 1,required
truste.cma.callApi('setConsentLevels', 'truste.com ', [1,2]);	0,1	permit 1,2 functional
truste.cma.callApi('setConsentLevels', 'truste.com ', [1,2,3,4,5]);	0,1,2,3,4	permit 1,2,3,4,5
truste.cma.callApi('setConsentLevels', 'truste.com ', [1,2,3]);	0,1,2	permit 1,2,3
truste.cma.callApi('setConsentLevels', 'truste.com ', [1,2]);	0,1	permit 1,2 functional
truste.cma.callApi('setConsentLevels',	0	permit 1,required

'truste.com ', [1]);		
truste.cma.callApi('setConsentLevels', 'truste.com ', [1,3]);	0,2	permit 1,3
truste.cma.callApi('setConsentLevels', 'truste.com ', [1,2,3]);	0,1,2	permit 1,2,3
truste.cma.callApi('setConsentLevels', 'truste.com ', [3);	0,2	permit 1,3
truste.cma.callApi('setConsentLevels', 'truste.com ', [1);	0	permit 1,required
truste.cma.callApi('setConsentLevels', 'truste.com ', [1,3);	0,2	permit 1,3