Customer Experience Digital Data Layer 1.0

Revision Date: December 10, 2013

Community Group Final Report

This version:

http://www.w3.org/2013/12/ceddl-201312

Authors:

Individual	Organization
Filip von Reiche	Acceleration
Miles Fender Murray Williams	Accenture
Reza Jalili	Adobe
Jay Myers	Best Buy
Jon Revill	Blue Cross and Blue Shield of North Carolina
Jason Walsh	BlueKai
Blane Sims Jared Vestal Eric Lunt	BrightTag
Paolo Margari	British Council
Patrick Wyatt	Criteo
Thomas Brune	Cubite GmbH
Sam Law	Conversion Foundry
Simon Rumble	Data Republic
Anna Long Eric Feinberg	Digital Analytics Association
Peter Loveday	Digital Window.
Fares Aldik	DIGITO Agency

Daniel Karpantschof	Economist Digital, The
Mark Prince Josh Goodwin Joe Orlet	Ensighten
Mathieu Jondet	Eulerian Technologies
Willem Paling	Foxtel
Laura Holmes Brian Kuhn Justin Cutroni Vishal Goenka Lukas Bergstrom	Google
Brian Hendrixson	HSN
Viswanath Srikanth Aubrey Rupp Hutch White	IBM
Casper Blicher Olsen	IIH Nordic
Amin Shawki	InfoTrust LLC
Andrew Thomas	Intelligent Research
Martijn van Berkum	GX Software
Lee Isensee	Localytics
Jonathan Weber	LunaMetrics
Gagan Kanwar Emilie Laffray William Hartley-Booth	Marin Software
Joseph Galarneau	Mezzobit
Aurelie Pols	Mind your Privacy S.L
Anton Gething	nToklo
Nicolas Malo	Optimal Ways
Roel Willems	Orange Valley Consultancy
Matt Stannard	4Ps Marketing
Rod Jacka	Panalysis

Harry Hurst Ian McCaig Stephen Elliott	QuBit
Keith Watkins	Red Hat
Jonathan Conway	Reevoo
Oliver Schiffers	SapientNitro
Alexander Dean	Snowplow Analytics
Adam Ware	SwellPath
Toby Doig Angus Glover Wilson	TagMan
Ali Behnam	Tealium
David Henderson	Triggered Messaging
Frederic Abrioux	wemoTech
David Evans	Whirlybird Consulting Limited
Evan Klein	Zaelab

Abstract

This specification describes a method for surfacing customer experience digital data on a web or other digital resource as a JavaScript Object which can be used for communicating this data to digital analytics and reporting servers.

Status of this Document

This specification was published by the Customer Experience Digital Data Community Group. It is not a W3C Standard nor is it on the W3C Standards Track. Please note that under the W3C Community Final Specification Agreement (FSA) other conditions apply. Learn more about W3C Community and Business Groups.

Table of Contents

1. Introduction

Collection and analysis of visitor behavioral and demographic data has become an integral part of web application design and website success, whether accessed through browsers on laptop, mobile, kiosk, tablet or another device. This data is central to site performance analysis, dynamically tailoring site content to visitor activity and interest and retargeting visitors based on their behaviors.

Increasingly, multiple vendors are involved in the data collection process for a given digital property, and each has a solution to be implemented on the page. As a result, page design has become more complex and development cycles have lengthened as different requirements for data surfacing and formatting are added to the implementation process. Further, changing or adding vendors sometimes requires that the development team change designs to accommodate vendor-specific requirements. Common data items must be continually surfaced in different ways, and each design requirement is a custom effort. Companies are searching for a simpler, more flexible, and standardized method to surface this common data across their digital properties.

This document details the specification for a standard data layer that collects this valuable user interaction information for subsequent use in analysis and reporting. The information in this document will be relevant to digital analysts and website implementation engineers, as well as to marketing professionals who need to understand user experience data that is being gathered.

2. Design Rationale and Goals

Customer experience digital data items that are tracked and captured by different vendors are commonly understood elements used in digital analytics, but vendor-specific format requirements and code assignments create design complexity and vendor-dependency in site design.

As a simple instance, Vendor A may capture some digital data for page details as a concatenation of `PageID + PAGENAME + PAGECATEGORY` while Vendor B may capture the same through distinct variables `vendorB.page.pageId`, `vendorB.page.pageName`, `vendorB.page.pageCategory`. Further, the names of the variables, the data structure name and hierarchy, and methods of extensibility all vary on a per-vendor basis. Frequently, custom code must be written to capture data to meet a given vendor's requirements.

Cumulatively, differences of this type between vendors permeate across all relevant customer experience digital data objects including pages, orders, shopping carts, registrants, and more, increasing the complexity of customer experience digital data management at the site.

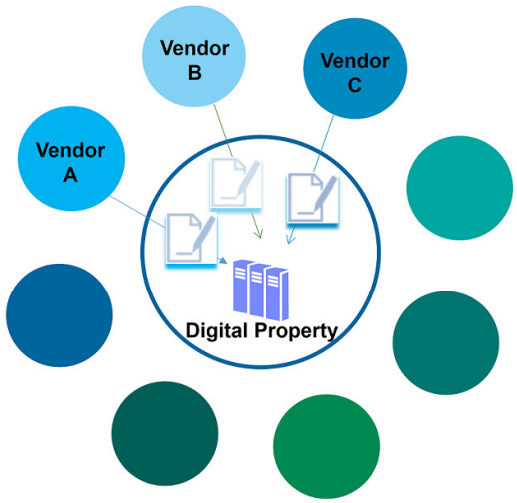


Figure 1: Vendor specific tags on a digital property

To this end, a standard data object that represents common data elements in a standardized way will allow development teams to implement design structures that populate the standard data object. Vendor code placed on the page would reference that standard structure, simplifying the process of onboarding or changing vendors, and shaving off expensive development cycles.

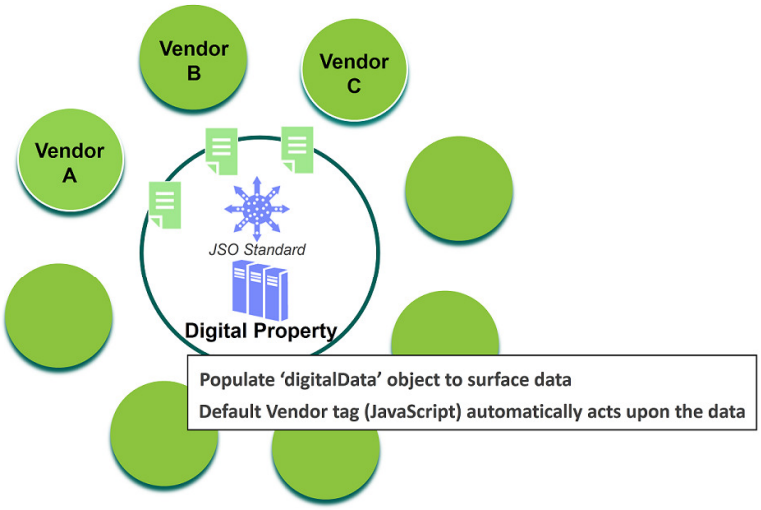


Figure 2: Common data object accelerates deployment, simplifies site management

The proposed standard data object is a JavaScript object because of ubiquitous support for JavaScript in web browsers and web-based applications, as well as in other forms of digital properties like mobile devices, kiosks, and other digital content.

The use of a JavaScript object means that the data is not embedded in the HTML markup and would not affect rendering of a page or performance. Developers will need only to populate the data fields in the object that are applicable to the page, keeping the size and complexity of the object to a minimum. This coding will never change regardless of vendor additions or deletions unless new data elements were required. Because the data is a standard object, vendors who recognize the object would provide code that references that object, requiring no modification of the page other than to drop the vendor code container onto the page.

This standard will thus yield a shorter development cycle using fewer resources and there will be a savings to the client in time and money. The addition or deletion of vendors will be independent of considerations of page redesign, allowing decisions to be made based on vendor efficacy rather than cost of migration.

3. How to Read this Document

To understand the value of the specification, Sections 1–3 give a high-level understanding behind the rationale and applicability of the specification. Section 4 shows examples of how code will be incorporated into your digital property, and Section 5 discusses privacy implications. Sections 6–7 dive into the specific objects and their properties.

Also note that **none of the objects in the specification are designed to be required** — use of each object is at the discretion of the digital property owner. However, once an object has been selected for use, the object must adhere to the structure as specified in this document to remain conformant.

The term **reserved** is encountered in the specification when defining Object Literals. In this context, names of certain values are reserved for specific use within an Object Literal, and these values should be populated only for the intended purpose to maintain the integrity of the specification.

The keywords MUST and REQUIRED, SHOULD and RECOMMENDED, MUST NOT, SHOULD NOT and NOT RECOMMENDED are to be interpreted according to RFC 2119.

4. Beginning with Examples

The specification relies on a JavaScript Object (JSO) to collect customer experience data. User interaction with a brand's digital property occurs in a wide variety of areas: customer service, e-commerce transactions, information portals, B2B partner interaction, and more, and the JSO is designed to accommodate these (as well as being extensible). Further sections of this document will delve into detailed specifications for the JSO, but following are simple examples of an instance of the JSO to illustrate its usage.

The JSO is designed to be contained within a root object called `digitalData` — this is a matter of convenience and gives a common starting point. All other objects are sub-object from this root object. There is a `pageInstanceID` that is used to identify the page being measured within a unique environment — development, staging, or production, for example. Beyond that, the specification includes sub-objects such as `page`, `product`, `cart`, `transaction`, `event`, `component`, `user`, `privacyAccessCategories`, and `version` for collecting different types of data in the JSO. (Additional objects can be added to `digitalData` as part of the extension mechanism.) Within the sub-objects, the specification defines a number of standard names for properties, while custom properties can also be added through an `attributes` object.

As an illustrative example, the `digitalData` object with the `page` object could be populated as below

```
digitalData = {
        pageInstanceID: "MyHomePage-Production",
        page:{
                pageInfo: {
                        pageID: "Home Page",
                        destinationURL: "http://mysite.com/index.html"},
                category:{
                        primaryCategory: "FAQ Pages",
                        subCategory1: "ProductInfo",
                        pageType: "FAQ"},
                attributes:{
                        country: "US",
                        language: "en-US"}
        }
  };
```

The specification allows the use of sub-objects and their properties as needed in any particular implementation. **While the presence of the sub-objects is optional, sub-objects that are populated must adhere to the syntax and semantics defined in this document to be conformant with the specification.**

Another instance of populating `digitalData` with both `page` and `product` objects is shown below:

```
digitalData = {

      pageInstanceID: "ProductDetailPageNikonCamera-Staging",
      page:{
            pageInfo:{
                  pageID: "Nikon Camera",
                  destinationURL:
                     "http://mysite.com/products/NikonCamera.html"},
            category:{
                  primaryCategory: "Cameras",
                  subCategory1: "Nikon",
                  pageType: "ProductDetail"},
            attributes:{
                  Seasonal: "Christmas"}
            },
      product:[{
            productInfo:{
                  productName: "Nikon SLR Camera",
                  sku: "sku12345",
                  manufacturer: "Nikon"},
            category:{
                  primaryCategory: "Cameras"},
            attributes:{
                  productType: "Special Offer"}
      }]
};
```

The `product` sub-object is an array with additional details to identify the product further.

As is evident from these examples, different pages that incorporate different types of data may populate different parts of the JSO to pass those data back to the server(s) for further analysis.

5. Privacy and Data Security Implications

Site visitor interactions with a website are collected as data, and could potentially include personally identifiable information or sensitive information about site visitors. The owner of a website is responsible for the legal consequences of the data collected, published in the site, and shared with technology vendors. This regulation differs from jurisdiction to jurisdiction and commonly takes the form of two classes of legislation:

- Site visitor privacy regulation (e.g., the EU Cookie Directive)
- Data security protection regulations (e.g., HIPAA)

Site visitor **privacy** regulations restrict site practices around whether or not to track a visitor's behavior. Data **security** protection regulations legislate whether a site owner can store and share visitor data, as well as how and for what duration you can retain that data (for more details on regulations, see Appendix B).

As the Customer Experience Digital Data initiative represents centralizing data collection, the JSO was designed with vendor-neutral objects that site owners can use to control which third-party technologies to enable on a site, and which data can be shared with the enabled technologies.

Leveraging the specification for privacy and security requires building the `privacy` object as well as the `security` objects for those components of data objects that are meant to be protected. Actual enforcement depends on processing that is external to the JSO. (See 6.10 and 6.11 for the specification of the `privacy` and `security` objects, and see Appendix B for recommendations to solution providers).

5.1 Privacy Object Categorizes Vendors

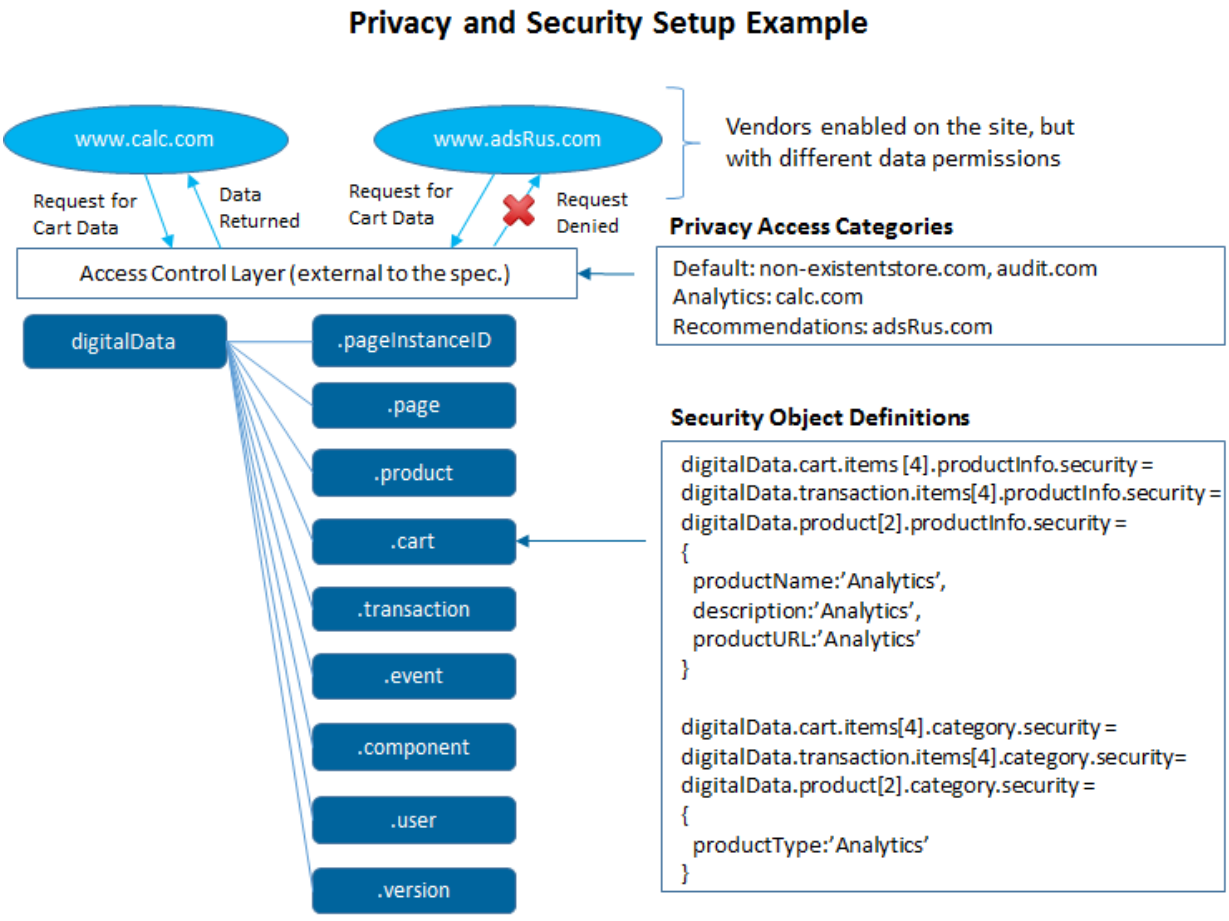
The JSO allows for the creation of a `privacy` object that is meant to categorize the various vendors with whom data may be shared into a small but meaningful set. The `privacy` object can be used to create vendor categories relevant for the website. For example, a site may create vendor categories such as "Analytics", "Personalization", and so forth. Categorizing the vendors in this fashion makes it easier to indicate what type of data sharing is appropriate with each.

5.2 Data Security property Categorizes Data

The JSO also allows for the creation of an optional `security` sub-object that is attached to any object in the specification. This `security` object identifies the categories of vendors (defined in the `privacy` object) with which that particular data property can be shared. For example, `productInfo.security = "Analytics"` implies that the `productInfo` property can be only shared with vendors categorized as "Analytics" vendors and vendors placed in the "Default" category in the `privacy` object.

5.3 Implementation Example

Consider the following example, of an e-commerce company, www.nonexistent-store.com, which uses tools from the vendors calc.com, adsRus.com, and audit.com, and uses the `digitalData` object below:



Because www.nonexistent-store.com operates primarily in the Netherlands, they have a requirement for privacy, and because they sell medication there are also concerns around the sensitivity of the data they collect and store.

To use the data standard, first they will categorize their technology vendors into categories:

- Default: nonexistent-store.com, audit.com
- Analytics: calc.com
- Recommendations: adsRus.com

To implement those categories, they will code the `digitalData.privacy.accessCategories` sub-object:

```
digitalData.privacy.accessCategories.categories[1] = {
      categoryName: "Default",
      domains: ["nonexistent-store.com", "audit.com"]};
digitalData.privacy.accessCategories.categories[2] = {
      categoryName: "Analytics",
      domains: ["calc.com"]};
digitalData.privacy.accessCategories.categories[3] = {
      categoryName: "Recommendations",
      domains: ["adsRus.com"]};
```

Second, they will identify any objects within the JSO that need data security, and add `security` objects for those objects. (The Default category is always allowed, because it represents the first party.) In this example, there is a need to protect parts of the `product`, `cart`, and `transaction` data from Recommendations category of tools because one product is sensitive. The following will restrict the data to only the “Analytics” and “Default” categories.

```
      digitalData.product[2].productInfo.security =
      digitalData.cart.item[4].productInfo.security =
      digitalData.transaction.item[4].productInfo.security = {
                productName: "Analytics",
                description: "Analytics",
                productUrl: "Analytics"};

      digitalData.product[2].category.security =
      digitalData.cart.item[4].category =
      digitalData.transaction.item[4].category = {
                productType: "Analytics"};
```

(Transaction data commonly contains information that could potentially identify a visitor; this example does not enumerate that configuration for brevity.)

Finally, the site will deploy a privacy enforcement technology, presumably via script tag at the top of the head in the page, which controls access to information in the `digitalData` object based on these categories. Recommendations to security solution providers are provided in Appendix B, as well as considerations for extending the specification in those specific implementations.

6. The Customer Experience Digital Data Object

This section carries the core specification.

Use of this specification can be adapted to specific cases, since the objects in the specification can be included or omitted as necessary or desired. None of the objects in this specification are required, but where objects are included they MUST conform to the Object Names and Types listed here.

6.1 The Root JavaScript Object

This section is normative

The root JavaScript Object (JSO) MUST be `digitalData`, and all data properties within this specification MUST fall within the hierarchy of the `digitalData` object.

The following sub-objects are defined as children of the `digitalData` object.

- `digitalData.pageInstanceID`
- `digitalData.page`
- `digitalData.product[n]`
- `digitalData.cart`
- `digitalData.transaction`
- `digitalData.event[n]`
- `digitalData.component[n]`
- `digitalData.user[n]`
- `digitalData.privacyAccessCategories`
- `digitalData.version`

The following subsections detail each of these sub-objects.

6.2 Page Identifier Object

This section is normative

The Page Identifier is among the most widely used web analytics data properties, and is among the top level `digitalData` objects. A Page Identifier, where included, MUST have the following Object Name & Type.

Object Name	Type
`digitalData.`**`pageInstanceID`**	String

A unique identifier for a page or other piece of content for which data is being collected.

This value SHOULD distinguish among environments, such as whether this page is in development, staging, or production. (Contrast with `digitalData.page.pageID` below, which may be unique only within a particular environment.)

6.3 Page Object

This section is normative

The Page object carries significant details about the page, and the most commonly used data elements are captured by the specification below. The Page object and its children, where included, MUST have the following Object Names & Types.

Object Name	Type
`digitalData.page`	Object Literal
Describes the page.	
`digitalData.page.pageInfo`	Object Literal
Describes details about the page.	

```
digitalData.page.pageInfo = {
          pageID: "316",
          pageName: "Rogaine Hair Regrowth Treatment",
          destinationURL: "http://site.com/r.html",
          referringURL: "http://www.google.com/url?&q=&esrc=s",
          sysEnv: "mobile",
          variant: "2",
          version: "1.14",
          breadCrumbs: ["home","Products","haircare"],
          author: "J Smith",
          issueDate: "2013-09-01",
          effectiveDate: "2013-09-20",
          expiryDate: "2014-09-20",
          language: "en-US",
          geoRegion: "US",
          industryCodes: "SIC codes",
          publisher: "BusDev"
     };
```

Reserved: `pageID` (String), `pageName` (String), `destinationURL` (String), `referringURL` (String), `sysEnv` (String), `variant` (String), `version` (String), `breadcrumbs` (Array::String), `author` (String), `issueDate` (String or Date Object), `effectiveDate` (String or Date Object), `expiryDate` (String or Date Object), `language` (String), `industryCodes` (String), `publisher` (String)

For `destinationURL`, and `referringURL`, RECOMMENDED values are `document.location`, and `document.referrer`, respectively.

The properties `onsiteSearchTerm` and `onsiteSearchResults` are RECOMMENDED for measuring the query term and number of results returned for onsite search functions.

For fields containing dates, JavaScript Date Objects or Strings using ISO 8601 are RECOMMENDED.

For fields containing language or locale, ISO 3166 and 639 are RECOMMENDED.

digitalData.page.**category**	Object Literal

Because of the wide range of methods for categorization, an object literal is provided for page categories.

```
digitalData.page.category = {
     primaryCategory: "FAQ Pages",
     subCategory1: "ProductInfo",
     pageType: "FAQ"
};
```

Reserved: `primaryCategory` (String)

The name `primaryCategory` is RECOMMENDED if you included only one set of categories for pages, or for your primary set of categories. All other names are optional and should fit the individual implementation needs in both naming and values passed.

digitalData.page.**attributes**	Object Literal

This object provides extensibility to the Page object. All names are optional and should fit the individual implementation needs in both naming and values passed.

6.4 Product Object

This section is normative

The Product object carries details about a particular product with frequently used properties listed below. This is intended for data about products displayed on pages or other content. For products added to a shopping cart or ordered in a transaction, see the Cart and Transaction objects below.

The Product object and its children, where included, MUST have the following Object Names & Types.

Object Name	Type
digitalData.**product[n]**	Array::Object Literal

List of products.

digitalData.product[n].**productInfo**	Object Literal

This object describes the product.

```
digitalData.product[n].productInfo = {
     productID: "rog3000",
     productName: "Rogaine",
     description: "Hair Regrowth",
     productURL: "http://site.com/r.html",
     productImage: "http://site.com/rog300_large.png",
     productThumbnail: "http://site.com/rog300_thumb.png",
     manufacturer: "Pharma",
     size: "300ml" };
```

Reserved: `productID` (String), `productName` (String), `description` (String), `productURL` (String), `productImage` (String), `productThumbnail` (String), `manufacturer` (String), `sku` (String), `color` (String), `size` (String)

All other names are optional and should fit the individual implementation needs in both naming and values passed.

`digitalData.product[n].`**`category`**	Object Literal

Because of the wide range of methods for categorization, an object literal is provided for product categories.

```
digitalData.product[n].category = {
     primaryCategory: "Haircare",
     subCategory1: "Men's",
     productType: "Thinning Hair Treatments"
};
```

Reserved: `primaryCategory` (String)

The name `primaryCategory` is RECOMMENDED if you included only one set of categories for products, or for your primary set of categories. All other names are optional and should fit the individual implementation needs in both naming and values passed.

`digitalData.product[n].`**`linkedProduct[n]`**	Array::Object Literal

List of products linked to this product.

`digitalData.product[n].linkedProduct[n].`**`productInfo`**	Object Literal

As for for each `linkedProduct`.

Reserved: `productID` (String), `productName` (String), `description` (String), `productURL` (String), `productImage` (String), `productThumbnail` (String), `manufacturer` (String), `sku` (string), `color` (String), `size` (String)

All other names are optional and should fit the individual implementation needs in both naming and values passed.

`digitalData.product[n].`**`attributes`**	Object Literal

This object provides extensibility to the Product object. Any additional dimensions related to the product can be provided. All names are optional and should fit the individual implementation needs in both naming and values passed.

6.5 Cart Object

This section is normative

The Cart object carries details about a shopping cart or basket and the products that have been added to it. The Cart object is intended for a purchase that has not yet been completed. See the Transaction object below for completed orders.

The Cart object and its children, where included, MUST have the following Object Names & Types.

Object Name	Type
digitalData.cart.**cartID**	String
An identifier for a particular shopping cart.	
digitalData.cart.**price**	Object Literal
This object provides details of the cart price. The basePrice SHOULD be the price of the items before applicable discounts, shipping charges, and tax. The cartTotal SHOULD be the total price inclusive of all discounts, charges, and tax.	
digitalData.cart.**attributes**	Object Literal
This object provides extensibility to the cart as a whole. Any additional dimensions related to the cart can be provided. All names are optional and should fit the individual implementation needs in both naming and values passed.	
digitalData.cart.**item[n]**	Array::Object Literal
List of items in the cart.	
digitalData.cart.item[n].**productInfo**	Object Literal
As for digitalData.product[n].productInfo, for each item.	

```
digitalData.cart.price = {
    basePrice: 200.00,
    voucherCode: "Alpha",
    voucherDiscount: 0.50,
    currency: "EUR",
    taxRate: 0.20,
    shipping: 5.00,
    shippingMethod: "UPS",
    priceWithTax: 120,
    cartTotal: 125
};
```

Reserved: basePrice (Number), voucherCode (String), voucherDiscount (Number), currency (String), taxRate (Number), shipping (Number), shippingMethod (String), priceWithTax (Number), cartTotal (Number)

For currency values, ISO 4217 is RECOMMENDED.

All other names are optional and should fit the individual implementation needs in both naming and values passed.

Reserved: `productID` (String), `productName` (String), `description` (String), `productURL` (String), `productImage` (String), `productThumbnail` (String), `manufacturer` (String), `sku` (string), `color` (String), `size` (String)

All other names are optional and should fit the individual implementation needs in both naming and values passed.

`digitalData.cart.item[n].`**`category`**	Object Literal

As for `digitalData.product[n].category`, for each `item`.

Reserved: `primaryCategory` (String)

The name `primaryCategory` is RECOMMENDED if you included only one set of categories for products, or for your primary set of categories. All other names are optional and should fit the individual implementation needs in both naming and values passed.

`digitalData.cart.item[n].`**`quantity`**	Number

Quantity of this particular item in the cart.

`digitalData.cart.item[n].`**`price`**	Object Literal

As for the main cart object `digitalData.cart.price`, where tracking on each individual item is desired.

Reserved: `basePrice` (Number), `voucherCode` (String), `voucherDiscount` (Number), `currency` (String), `taxRate` (Number), `shipping` (Number), `shippingMethod` (String), `priceWithTax` (Number)

All other names are optional and should fit the individual implementation needs in both naming and values passed.

`digitalData.cart.item[n].`**`linkedProduct[n]`**	Array::Object Literal

List of products linked to this item in the cart, as for `digitalData.product[n].linkedProduct[n]`, for each of the `item`.

`digitalData.cart.item[n].linkedProduct[n].`**`productInfo`**	Object Literal

As for `digitalData.productInfo`, for each `linkedProduct`.

Reserved: `productID` (String), `productName` (String), `description` (String), `productURL` (String), `productImage` (String), `productThumbnail` (String), `manufacturer` (String), `sku` (string), `color` (String), `size` (String)

All other names are optional and should fit the individual implementation needs in both naming and values passed.

`digitalData.cart.item[n].`**`attributes`**	Object Literal

This object provides extensibility to each item within the cart. Any additional dimensions related to the item can be provided. All names are optional and should fit the individual implementation needs in both naming and values passed.

6.6 Transaction Object

This section is normative

The Transaction object is similar to the Cart object, but represents a completed order. The Transaction object contains analogous sub-objects to the Cart object as well as additional sub-objects specific to completed orders.

The Transaction object and its children, where included, MUST have the following Object Names & Types.

Object Name	Type
digitalData.transaction.**transactionID**	String
A unique identifier for a particular transaction; usually an existing order identifier.	
digitalData.transaction.**profile**	Object
A profile for information about the purchaser, typically associated with a registered user.	
digitalData.transaction.profile.**profileInfo**	Object Literal
An extensible object for providing information about the purchaser.	

```
digitalData.transaction.profile.profileInfo = {
    profileID: "humanbeing12345",
    userName: "me"
};
```

Reserved: `profileID` (String), `userName` (String), `email` (String)

All other names are optional and should fit the individual implementation needs in both naming and values passed.

digitalData.transaction.profile.**address**	Object Literal
An extensible object for providing (billing) address information for the purchaser.	

```
digitalData.transaction.profile.address = {
    line1: "673 My Street",
    line2: "Apt 1",
    city: "Austin",
    stateProvince: "TX",
    postalCode: "78610",
    country: "USA"
};
```

Reserved: `line1` (String), `line2` (String), `city` (String), `stateProvince` (String), `postalCode` (String), `country` (String),

All other names are optional and should fit the individual implementation needs in both naming and values passed.

digitalData.transaction.profile.**shippingAddress**	Object Literal

As for `address`, but for the shipping address.

Reserved: `line1` (String), `line2` (String), `city` (String), `stateProvince` (String), `postalCode` (String), `country` (String),

All other names are optional and should fit the individual implementation needs in both naming and values passed.

digitalData.transaction.**total**	Object Literal

This object provides details of the final price the purchaser has to pay. The `basePrice` SHOULD be the price of the `item`s before applicable discounts, shipping charges, and tax. The `transactionTotal` SHOULD be the total price inclusive of all discounts, charges, and tax.

```
digitalData.transaction.total = {
     basePrice: 200.00,
     voucherCode: "Alpha",
     voucherDiscount: 0.50,
     currency: "EUR",
     taxRate: 0.20,
     shipping: 5,
     shippingMethod: "UPS",
     priceWithTax: 120,
     transactionTotal: 125
};
```

Reserved: `totalPrice` (Number), `voucherCode` (String), `voucherDiscount` (Number), `currency` (String), `taxRate` (Number), `shipping` (Number), `shippingMethod` (String), `priceWithTax` (Number), `transactionTotal` (Number)

All other names are optional and should fit the individual implementation needs in both naming and values passed.

digitalData.transaction.**attributes**	Object Literal

This object provides extensibility to the `transaction` as a whole. Any additional dimensions related to the cart can be provided. All names are optional and should fit the individual implementation needs in both naming and values passed.

digitalData.transaction.**item[n]**	Array

List of items in the transaction.

digitalData.transaction.item[n].**productInfo**	Object Literal

As for `digitalData.product[n].productInfo`, for each item.

The `productID` property is RECOMMENDED.

Reserved: `productID` (String), `productName` (String), `description` (String), `productURL` (String), `productImage` (String), `productThumbnail` (String), `manufacturer` (String), `sku` (string), `color` (String), `size` (String)

All other names are optional and should fit the individual implementation needs in both naming and values passed.

`digitalData.transaction.items[n].`**`category`**	Object Literal

As for `digitalData.product[n].category`, for each item.

Reserved: `primaryCategory` (String)

All other names are optional and should fit the individual implementation needs in both naming and values passed.

`digitalData.transaction.item[n].`**`quantity`**	Number

Quantity of this particular item ordered.

`digitalData.transaction.item[n].`**`price`**	Object Literal

As for the main transaction object `digitalData.transaction.price`, where tracking on each individual item is desired.

Reserved: `basePrice` (Number), `voucherCode` (String), `voucherDiscount` (Number), `currency` (String), `taxRate` (Number), `shipping` (Number), `shippingMethod` (String), `priceWithTax` (Number)

All other names are optional and should fit the individual implementation needs in both naming and values passed.

`digitalData.transaction.item[n].`**`linkedProduct[n]`**	Array::Object Literal

List of products linked to this item in the order, as for `digitalData.product[n].linkedProduct[n]`, for each item.

`digitalData.transaction.item[n].linkedProduct[n].`**`productInfo`**	Object Literal

As for `digitalData.productInfo`, for each `linkedProduct`.

Reserved: `productID` (String), `productName` (String), `description` (String), `productURL` (String), `productImage` (String), `productThumbnail` (String), `manufacturer` (String), `sku` (string), `color` (String), `size` (String)

All other names are optional and should fit the individual implementation needs in both naming and values passed.

`digitalData.cart.items[n].`**`attributes`**	Object Literal

This object provides extensibility to each item within the transaction. Any additional dimensions related to the item can be provided. All names are optional and should fit the individual implementation needs in both naming and values passed.

6.7 Event Object

This section is normative

The Event object collects information about an interaction event by the user. An event might be a button click, the addition of a portal widget, playing a video, adding a product to the shopping cart, etc. Any action on the page could be captured by an Event object.

The Event object and its children, where included, MUST have the following Object Names & Types.

Object Name	Type
`digitalData.event[n]`	Array

List of events that were measured.

`digitalData.event[n].eventInfo`	Object Literal

This object describes the event.

```
digitalData.event[n].eventInfo = {
    eventName: "Add News Portal",
    eventAction: "addportal",
    eventPoints: 200,
    type: "contentModifier",
    timeStamp: new Date(),
    effect: "include portal 1234"
};
```

The `eventInfo` object MUST include `eventName`.

Reserved: `eventName` (String), `eventAction` (String), `eventPoints` (Number), `type` (String), `timeStamp` (String or Date Object), `cause` (String), `effect` (String)

All other names are optional and should fit the individual implementation needs in both naming and values passed.

`digitalData.event[n].category`	Object Literal

Because of the wide range of methods for categorization, an object literal is provided for page categories.

```
digitalData.event[n].category = {
    primaryCategory: "Portal",
    subCategory1: "dashboard"
};
```

Reserved: `primaryCategory` (String)

The name `primaryCategory` is RECOMMENDED if you included only one set of categories for events, or for your primary set of categories. All other names are optional and should fit the individual implementation needs in both naming and values passed.

digitalData.event[n].**attributes**	Object Literal

This object provides extensibility to each event. Any additional dimensions related to the event can be provided. All names are optional and should fit the individual implementation needs in both naming and values passed.

6.8 Component Object

This section is normative

The Component object is intended to capture information about a content component included as part of a page, such as a video. Interactions with the component — such as playing the video — would be an Event, captured by the Event object above.

The Component object and its children, where included, MUST have the following Object Names & Types.

Object Name	Type
digitalData.**component[n]**	Array::Object Literal

List of page components.

digitalData.component[n].**componentInfo**	Object Literal

This object describes the component.

```
digitalData.component[n].componentInfo = {
     componentID: "rog300v",
     componentName: "How to Use Rogaine",
     description: "Hair Treatment Video"
};
```

Reserved: componentID (String)

All other names are optional and should fit the individual implementation needs in both naming and values passed.

digitalData.component[n].**category**	Object Literal

Because of the wide range of methods for categorization, an object literal is provided for page categories.

```
digitalData.component[n].category = {
     primaryCategory: "Haircare",
     subCategory1: "Videos",
     componentType: "Flash Movie"
};
```

Reserved: primaryCategory (String)

The name `primaryCategory` is RECOMMENDED if you included only one set of categories for components, or for your primary set of categories. All other names are optional and should fit the individual implementation needs in both naming and values passed.

`digitalData.component[n].`**`attributes`**	Object Literal

This object provides extensibility to each component. Any additional dimensions related to the component can be provided. All names are optional and should fit the individual implementation needs in both naming and values passed.

6.9 User Object

This section is normative

The User object captures the profile of a user who is interacting with the website.

The User object and its children, where included, MUST have the following Object Names & Types.

Object Name	**Type**
`digitalData.`**`user[n]`**	Array::Object Literal

List of user(s) interacting with the page. (Although typically web data has a single user per recorded interaction, this object is an array and can capture multiple users.)

`digitalData.user[n].`**`segment`**	Object Literal

This object provides population segmentation information for the user, such as premium versus basic membership, or any other forms of segmentation that are desirable. Any additional dimensions related to the user can be provided. All names are optional and should fit the individual implementation needs in both naming and values passed.

`digitalData.user[n].`**`profile[n]`**	Array::Object Literal

A profile for information about the user, typically associated with a registered user. (Although typically a user might have only a single profile, this object is an array and can capture multiple profiles per user.)

`digitalData.user[n].profile[n].`**`profileInfo`**	Object Literal

An extensible object for providing information about the user.

```
digitalData.transaction.profile.profileInfo = {
     profileID: "humanbeing12345",
     userName: "me"
};
```

Reserved: `profileID` (String), `userName` (String), `email` (String), `language` (String), `returningStatus` (String), `type` (String)

All other names are optional and should fit the individual implementation needs in both naming and values passed.

digitalData.user[n].profile[n].**address**	Object Literal

An extensible object for providing address information for the user.

```
digitalData.transaction.profile.address = {
      line1: "673 My Street",
      line2: "Apt 1",
      city: "Austin",
      stateProvince: "TX",
      postalCode: "78610",
      country: "USA"
};
```

Reserved: `line1` (String), `line2` (String), `city` (String), `stateProvince` (String), `postalCode` (String), `country` (String),

All other names are optional and should fit the individual implementation needs in both naming and values passed.

digitalData.user[n].profile[n].**social**	Object Literal

An extensible object for providing social information for the user profile.

```
digitalData.user[n].profile[n].social = {
      twitter: "somebody",
      twitterInfo: "stuff",
      facebook: "somebody1234",
      facebookInfo: "morestuff"
};
```

All names are optional and should fit the individual implementation needs in both naming and values passed.

digitalData.user[n].profile[n].**attributes**	Object Literal

This object provides extensibility to the profile.

All other names are optional and should fit the individual implementation needs in both naming and values passed.

6.10 Privacy Object

This section is normative

The Privacy object holds the privacy policy settings that could be used to:

1. Capture and enforce site visitor consent to use tracking technologies on the site.
2. Together with Security objects described below, secure access to individual objects within the JSO by categories of tracking technologies.

The Privacy object and its children, where included, MUST have the following Object Names & Types.

Object Name	Type
digitalData.privacy.**accessCategories[n]**	Array::Object Literal

A list of privacy categories.

digitalData.privacy.accessCategories[n].**categoryName**	String

Definition of category names to be associated with `security` objects with the matching name.

A `categoryName` of Default MUST be included for privacy enforcement. All other categories are custom. Common categories include Analytics, Recommendations, and Personalization.

digitalData.privacy.accessCategories[n].**domains**	Array

Particular vendors associated with the particular category. This SHOULD be an array listing domains for vendors associated with the category.

The Default category SHOULD contain all technologies which must be treated the same as first party for privacy purposes (e.g., a tag management technology).

6.11 Security Object

This section is normative

The Security object is an optional sub-object of each object in the JSO, which can be used to enforce data access control over that object. If a Security object is defined for an object, the value should be a comma-separated list of categories defined as `categoryName` in the Privacy object. If a Security object is not defined for an object, no data access controls will be enforced.

The Default category is required for security enforcement. The Default category has access to all children of the digitalData element and does not need to be explicitly listed in Security objects.

For instance, the security object associated with `cartID` would be written as:

Object being secured: `digitalData.cart.cartID`
Security object: `digitalData.cart.cartID.security: "Analytics"`

For object literals, the same structure used in the the object literal is repeated in the Security object with values for the security levels. For instance:

Object being secured:

```
digitalData.transaction.profile.shippingAddress = {
      line1: "673 Mystreet",
      line2: "Apt 1",
      city: "Austin",
      stateProvince: "TX",
      postalCode: "78610",
      country: "USA",
      shippingMethod: "UPS",
      specialInstruction: "Beware of Dog"
};
```

Security object:

```
digitalData.transaction.profile.shippingAddress.security = {
      line1: "Personalization",
      line2: "Personalization",
      city: ["Analytics","Personalization"],
      stateProvince: "Analytics",
      postalCode: "Analytics"
};
```

Because `shippingMethod` and `specialInstruction` do not have a security object defined, there is no access control on those elements.

Object Name	Type
`*.security`	String or Object Literal, depending on the parent object being secured.

Security sub-object that captures the specific setting for its parent object. The value of security properties MUST be a string or array with values drawn from the values of `digitalData.privacy.accessCategories[n].categoryName`.

An empty string for a Security object indicates that only the Default category is allowed to access the data. No Security object denotes that no data security protection is necessary for a piece of data.

Security object with empty string: `digitalData.cart.cartID.security = "";`

Only the Default category can access the `cartID` property. Synonymous with `digitalData.cart.cartID.security = "Default";`.

Security object not specified: `digitalData.cart.cartID.security = null;`

Any category can access the cartID property (i.e., no data access controls are enforced).

6.12 Version Object

This section is normative

The version of this specification used by the `digitalData` object.

Object Name	Type
`digitalData.version`	String

The version number corresponding to the W3C Community Specification for Customer Experience Digital Data followed by the JSO. For `digitalData` objects conforming to this specification document, the value SHOULD be `digitalData.version = "1.0"`.

6.13 Extending the Specification

Extending this specification is straightforward: implementers can add appropriate sub-objects or properties as needed.

Adding a new object. A new object can be added to the digitalData object to represent a completely new type of data. This could be added as a first-level object, such as the following:

```
digitalData.newObject = { };
```

It could also take the form of adding a new sub-object to an existing object:

```
digitalData.transaction.newObject = { };
```

Adding a new properties. An existing object can be extended by adding additional name-value pairs:

```
digitalData.user[n].profile[n] = { newValue: "value" };
```

If reserved names are employed, they MUST use the types specified. If other names are employed, the values may take on any type. All of the first-level objects and some of their children contain an `attributes` object for extensibility, which does not reserve any names.

When extending the data layer, names SHOULD NOT use JavaScript reserved words, predefined names, or event handlers for the names of objects or properties. These are laid out in Appendix A.

7. Industry Specific Examples of Using the Specification

The specification, as shown above, is easily extensible by adding new objects, sub-objects, or additional name-value properties to object literals. Thus the specification can be used across industries by either adding entirely new objects, or extending existing objects in the specification.

7.1 Example from Air Travel Industry

Following are a number of possible examples of employing and extending this specification for data for an air travel booking website.

- Core flight reservation related data can be used to populate and extend the `productInfo` field as shown below:

```
digitalData.product[n].productInfo = {
      productID: "734565538989889110",
      description: "Business Class One-Way Ticket",
      originAirportCode: "RDU",
      originCity: "Raleigh",
      originState: "North Carolina",
      originCounty: "USA",
      destinationAirportCode: "BOM",
      destinationState: "Maharashtra",
      destinationCountry: "India",
      departureDate: new Date("December 15, 2013 14:20:00"),
      arrivalDate: new Date("December 16, 2013 21:40:00"),
      numberOfTravellers: 1
};
```

- As a travel product is moved into the cart details such as price, fees, and currency can be used to populate or extend the `digitalData.cart.price` object literal. Additional fields such as a confirmation number might extend the `digitalData.transaction` objects.

```
digitalData.cart.price = {
      basePrice: 1000.00,
      currency: "USD",
      fees: 200,
      taxRate: 0.08,
      cartTotal: 1296.00
};
```

- If a visitor carries out an on-site search for available flights, the search and its results can be captured using the existing `digitalData.page.pageInfo.onsiteSearchTerm` and `digitalData.page.pageInfo.onsiteSearchResult` objects.
- Details regarding the Customer making the purchase can be captured in the User object. Existing objects in the specification could be used, such as:
    - `digitalData.user[n].profile[n].address`, for the customer's postal address.

- `digitalData.user[n].profile[n].profileInfo` for any other customer identifiers.
- A "frequent flyer club" property could be added to the `digitalData.user[n].segment` object to capture the customer's loyalty level:

```
digitalData.user[n].segment = {
      frequentFlyerClub: "Silver Elite"
      };
```

7.2 Example from Healthcare Insurance Industry

As another example, to collect data in the healthcare insurance industry, it might be more convenient to model a few entirely new objects to capture relevant information.

One such bespoke object could be `digitalData.member`, used to capture a member' s account details with the institution, which may frequently be insurance related. This could include sub-objects as detailed below:

Example Member object

```
digitalData.member[n] = {
      memberID: "2723 49202388 01",
      age: "41",
      groupRelationship: "436378",
      groupName: "Employer\, Inc.",
      relation: "Spouse",
      gender: "M",
      originalJoinDate: "2011-01-21",
      postalCode: "15214"
};
```

Other useful objects may include an Application object and a Plan object, with additional sub-objects under each.

Example Application object

```
digitalData.application = {
      appID: "7565-2373-0086-8937",
      source: "Telephone",
      status: "Pending",
      creationDate: new Date("December 15, 2013 14:20:02"),
      completionDate: new Date("December 15, 2013 16:05:16")
};
```

Example Plan object

```
digitalData.plan = {
      name: "Family Advantage 250",
      type: "EPO",
      policyStatus: "Current",
      premium: 454.25,
      effectiveDate: new Date("December 15, 2013 16:05:16") };
```

8. Acknowledgements

The editors would like to acknowledge, in addition to all the mentioned authors, the contributions of Eliot Towb, Michael Niemann, Jim Colson, Christopher B Ferris, Tom Glover, Karen Myers for giving this initiative the initial momentum and contributions to the specification.

9. References

RFC 2119: *Key words for use in RFCs to Indicate Requirement Levels* , S. Bradner, Author. Internet Engineering Task Force, March 1997. Available at http://www.ietf.org/rfc/rfc2119.txt.

RFC 4627: *The application/json Media Type for JavaScript Object Notation (JSON), D. Crockford,* Author. Internet Engineering Task Force, July 2006. Available at http://www.ietf.org/rfc/rfc4627.txt.

Appendices

A. Reserved Names and Identifiers

Avoid the following Reserved Words, Predefined Names, and Event Handlers for the object or property names when extending the specification.

Reserved Words:

break	enum	instanceof	super	yield
case	export	interface	switch	
catch	extends	let	this	
class	finally	new	throw	
continue	for	package	try	
debugger	function	private	typeof	
default	if	protected	var	
delete	implements	public	void	
do	import	return	while	
else	in	static	with	

Predefined Names:

alert	element	Infinity	Object	scroll
all	elements	isFinite	offscreenBuffering	secure
anchor	embed	isNaN	open	select
anchors	embeds	isPrototypeOf	opener	self
area	encodeURI	java	option	setInterval

	encodeURIComponent	JavaArray	outerHeight	setTimeout
assign	escape	JavaClass	outerWidth	status
blur	eval	JavaObject	packages	String
button	event	JavaPackage	pageXOffset	submit
checkbox	fileUpload	innerHeight	pageYOffset	taint
clearInterval	focus	innerWidth	parent	text
clearTimeout	form	layer	parseFloat	textarea
clientInformation	forms	layers	parseInt	top
close	frame	length	password	toString
closed	frames	link	pkcs11	undefined
confirm	frameRate	location	plugin	unescape
constructor	function	Math	prompt	untaint
crypto	getClass	mimeTypes	propertyIsEnum	valueOf
Date	hasOwnProperty	name	prototype	window
decodeURI	hidden	NaN	radio	
decodeURIComponent	history	navigate	reset	
defaultStatus	image	navigator	screenX	
document	images	Number	screenY	

Event Handlers:

onbeforeunload	oncontextmenu	onkeydown	onmousedown	onmouseup
onblur	ondragdrop	onkeypress	onmousemove	onreset
ondragdrop	onerror	onkeyup	onmouseout	onsubmit
onclick	onfocus	onload	onmouseover	onunload

Examples of identifiers to avoid include:

```
digitalData.user.public
digitalData.form
digitalData.transaction.status
```

B. Privacy — Additional Considerations

B.1 Recommendations to Privacy and Security Solution Providers

The Privacy and Security objects were designed to be vendor and technology neutral, while providing all of the necessary data for functional enforcement without vendor lock-in. The following sections provide detail on how to enforce, and allow for open source and home-grown solutions, as needed.

B.1.1 Privacy

Technology providers can build support for this standard by enforcing the configured preference. A technology which supports this standard must:

1. Provide interface to gather visitor consent for categories of technologies.
2. Enforce the visitor consent preference, and thereby prevent the execution of sharing technologies or prevent the sharing of data by technologies

As of September 2013, there are at least three current approaches to address this concern:

1. Conditional Logic -- These technologies use a helper function to wrap every 3rd party JavaScript in conditional logic, governed by interface controls. For example: github.com/creativeaura/eu-cookie-opt-in
2. Firewall -- This technology deploys JavaScript to the browser which blocks JavaScript from being added if the site visitor has opted out. For example: developers.google.com/analytics/devguides/collection/analyticsjs/advanced#optout
3. Uncookieing or Unpixeling -- These technologies send a request to individual or all technology networks to opt-out from data collection. The technology networks set a third party cookie (pixel) indicating that the site visitor should not be tracked. Subsequent page loads are ignored by technology networks. For example: aboutads.info/choices/

B.1.2 Data Security

Technology providers can support this standard by providing the functionality to execute controlled data sharing as defined by the website administrator. A technology which supports this standard must:

1. Provide an interface for website owners to create privacy objects and security sub-objects. This interface must include the flexibility to define privacy categories and names since these can vary greatly across industries and organizations.
2. Some method of supporting de-identification or anonymization algorithms, either built into the tool, through API-type call, or some external access.

B.2 Recommendations for Enhancing Privacy and Data Security

B.2.1 Privacy

A fellow subgroup of the W3C is working on an open standard by which web users can express their desire to opt in or out of tracking technologies (Tracking Preference Expression, under development by the W3C Tracking Protection Working Group). As that standard develops and gains adoption, we hope and expect Privacy technologies will seek to be compatible with that specification.

B.2.2 Data Security

Data can be protected by restricting which vendors it is shared with, as intended by this standard, and by sharing it in appropriate privacy-protecting formats (add note for reference with case studies). There is a clear place to build additional specification for data handling policies to govern this sharing -- anonymization, data-retention, de-identification, deletion, encryption, to extend the data access controls enabled by this specification. The W3C Tracking Protection Working Group is also developing a standard to define compliance to users' tracking preferences (Tracking Compliance and Scope). With adoption of that specification as a standard, we hope and expect privacy technologies will support it as well.

B.3 Privacy Regulations

B.3.1 Site Visitor Privacy Regulations

Countries have enacted legislation requiring that site owners offer site visitors the ability to opt-out from or opt-in to having their interactions tracked. This class of regulation is commonly referred to as "cookie laws", due to the general emphasis on cookies, one method of sharing data across organizations. As a result of these laws, site owners have implemented in-house or commercial solutions to address these concerns. As of September 2013, many European Union countries have enacted legislation based on a 2011 European Union directive. Other regions outside the EU are also considering or enacting legislation in this area. For instance, Mexico and New Zealand already have legislation in this regard.

You need to identify the legislation impacting your site, prior to implementing a Privacy solution. The local legislation will indicate whether visitors must have the ability to opt-out, or whether visitors must consent to opt-in for tracking. This determines the interface options you can use, and whether a site owner can track initial page views.

You should also understand which technologies deployed on your site share data, as these are the technologies that regulations are enacted to restrict.

B.3.2 Data Security Protection Regulations

When you understand the data your site is collecting and the regulations that apply to it, you need to develop your security approach. The following questions address several common dimensions of security:

- How will you protect personally identifiable information (which may itself be defined differently based on regulations) or other sensitive data such as bank account or insurance subscriber numbers?

Common policies are to restrict data access, delete the data, encrypt the data, or de-identify the data. At a minimum you will want to restrict who can access the data. As such, this standard provides a way to configure access controls over the data. Other options are to delete, encrypt the data, or even de-identify or anonymize the data -- alter it in some way so that the data owners cannot be easily identified.

- How long and under what circumstances is this data retained?

  Retention periods for individual data should be set as well as decisions about aggregation and archiving.

- Is an oversight program a good fit for your website? Several organizations, such as TRUSTe, the Digital Advertising Alliance, and the Network Advertising Initiative, have programs providing privacy oversight for website administrators. These organizations develop tracking and privacy practices that they expect their members to adhere to. Most of these programs are tailored to online advertising, where there is the greatest threat of sharing.

You should consider these aspects of security as you develop your plan for securing your data.