A Standard Data Layer for Customer Experience Data

Abhishek Tiwari

Citation: *A. Tiwari*, "A Standard Data Layer for Customer Experience Data", Abhishek Tiwari, 2013. doi:10.59350/k2ycj-0gq25

Published on: December 25, 2013

After much deliberation, recently W3C Customer Experience Digital Data Community Group released version 1.0 of Customer Experience Digital Data Layer (CEDDL). This specification is designed to communicate customer experience digital data to analytics and reporting services.

What is a data layer?

A data layer is a standard way to format data within a web page. It is collection of one or more JavaScript Objects (JSO) holding information or important signals about page and user.

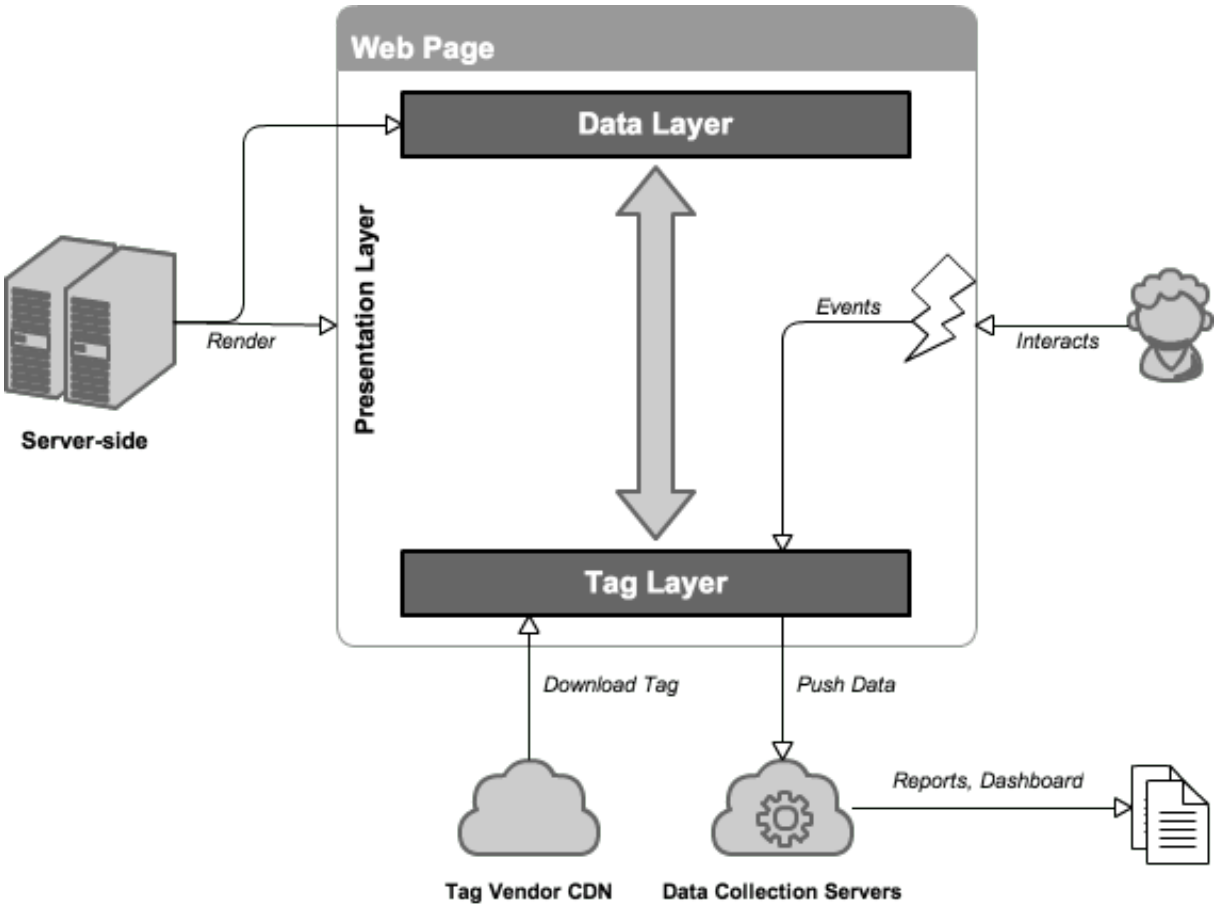

Figure 1: A data layer separates presentation layer and tag layer from data.

A data layer decouples analytics and marketing tags from data. It separates presentation layer of the page from data without affecting rendering of a page or performance. In addition, data layer enables developer to implement clean and highly maintainable tracking code.

It also makes data reusable and act as single source of truth for all analytics and marketing tags. Using

data layer tags can communicate and share data on the web page. Application of data layer is not limited to marketing or analytics tools. Information in the data layer can then be consumed by any type of web application.

Data Source

As described earlier, a data layer holds information or important signals about page and user. Hence, a data layer is quite dynamic in nature. Actual state of data layer varies per-page and per-user basis. That said, a data layer has two primary data sources,

1. Data received from server-side (core data page and user)
2. Data appended by tag layer or client-side JavaScript (cookies, events, page attributes, etc.)

Why we need a data layer standard?

If you familiar with any tag management system (TMS) like Google Tag Manager(GTM), Tealium, BrightTag, Ensighten, etc., then data layer may not be surprise for you. Unfortunately each tag management system follows its own data layer convention. Time and money aside, this creates design complexity, maintenance and vendor lock-in issues.

Here's an example of how an ecommerce transaction would look using Google Tag Manager data layer,

```
<script>
var dataLayer = [{
    'transactionId': '1234',
    'transactionTotal': 32.00,
    'transactionTax': 0.20,
    'transactionShipping': 5,
    'transactionProducts': [{
        'sku': 'DD44',
        'name': 'T-Shirt',
        'category': 'Apparel',
        'price': 15.99,
        'quantity': 1
    },{
        'sku': 'AA1243544',
        'name': 'Socks',
        'category': 'Apparel',
        'price': 9.99,
        'quantity': 1
    }]
}];
</script>
```

This is equivalent ecommerce transaction using Tealium Universal Data Object, ~~~

```
As you can see from above examples, names of the variables, data structure
     name (`data_layer` vs `utag_data`) and hierarchy (flat vs nested), etc
    ., vary per-vendor basis.

# A Standard Data Layer
Customer Experience Digital Data Layer (CEDDL) is very first community
    effort to create a standard data layer. A standard data layer will
    yield savings in terms of time and money.

This means data layer implementation will remain independent of vendors.
    Adding or removing vendor tags will be straightforward.  A vendor tag
    will reference standard data structure.

![A standard data layer for vendor independent customer experience data.
    Specification relies on a JavaScript Object (JSO) to collect customer
    experience data.](https://static.abhishek-tiwari.com/old-ghost/images/
    standard-data-layer-for-customer-experince.png "A standard data layer
    for vendor independent customer experience data. Specification relies
    on a JavaScript Object or JSO to collect customer experience data.")

# Core Specification
In CEDDL, customer experience data is collected in a root JavaScript
    Object (JSO) named as `digitalData`. This root object contains sub-
    objects `page`, `product`, `cart`, `transaction`, `event`, `component`,
     `user`, `privacy`, and `version`. These sub-objects collect different
    types of data in the JSO. Every `digitalData` object has a `
    pageInstanceID` that is used to identify the page being measured within
     a unique environment – development, staging, or production. Objects `
    product`, `cart` and `transaction` are mainly relevant for an ecommerce
     website.

Each sub-object contains pre-defined properties. In addition, custom
    properties can be added using `attributes` object. Depending on page
    and user, different parts of `digitalData` object will be populated
    which means based on context sub-object and properties will be optional
    .

A common way to declare `digitalData` object can be,
```

digitalData = { pageInstanceID: “PageIdentifier-Environment”, subObject1: {key-value pairs}, subObject2: {key-value pairs}, subObject3: {subObject3: {key-value pairs},}, }; ~~~

Alternatively, you can also declare `digitalData` using this syntax,

```
digitalData.pageInstanceID = "PageIdentifier-Environment";
digitalData.subObject1= {key-value pairs};
digitalData.subObject2= {key-value pairs};
digitalData.subObject3.subObject4= {key-value pairs};
```

So for instance, on a product page we may populate information about `page` and `product` together or separately.

```
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"}
       }]
 };
```

Here is a more granular alternative version,

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

```
        }];
```

Once an order is successfully placed, on confirmation page we can just populate `transaction` object.

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

Extensibility

Extending CEDDL is quite easy — add appropriate sub-objects and/or properties. When adding new objects or properties one should not use JavaScript keywords.

```
//Adding a new object to root object
digitalData.newObject = { };
//Adding a new sub-object to an existing sub-object
digitalData.transaction.newObject = { };
//Adding a new property
digitalData.user[n].profile[n] = { newValue: "value" };
```

Privacy & Security

In many countries privacy laws require that website give visitor options like,

- what's being tracked about them,
- ability to opt-in or opt-out of being tracked

In addition, sometime we want to enforce data access controls on per-vendor basis or based on vendor category. For instance, Google Analytics prohibits sending personally identifiable information (PII).

In following illustration, we first declared 3 `digitalData.privacy.accessCategories` sub-objects with `categoryName` as `Default`, `Analytics` and `Recommendations`. Each `digitalData.privacy.accessCategories` contains a list of `domains` for vendors associated with the `categoryName`. Then we identified any objects within the JSO that need

data security, and added security sub-objects under those objects. Security sub-object consists a comma-separated list of categories defined as `categoryName` in the Privacy object.

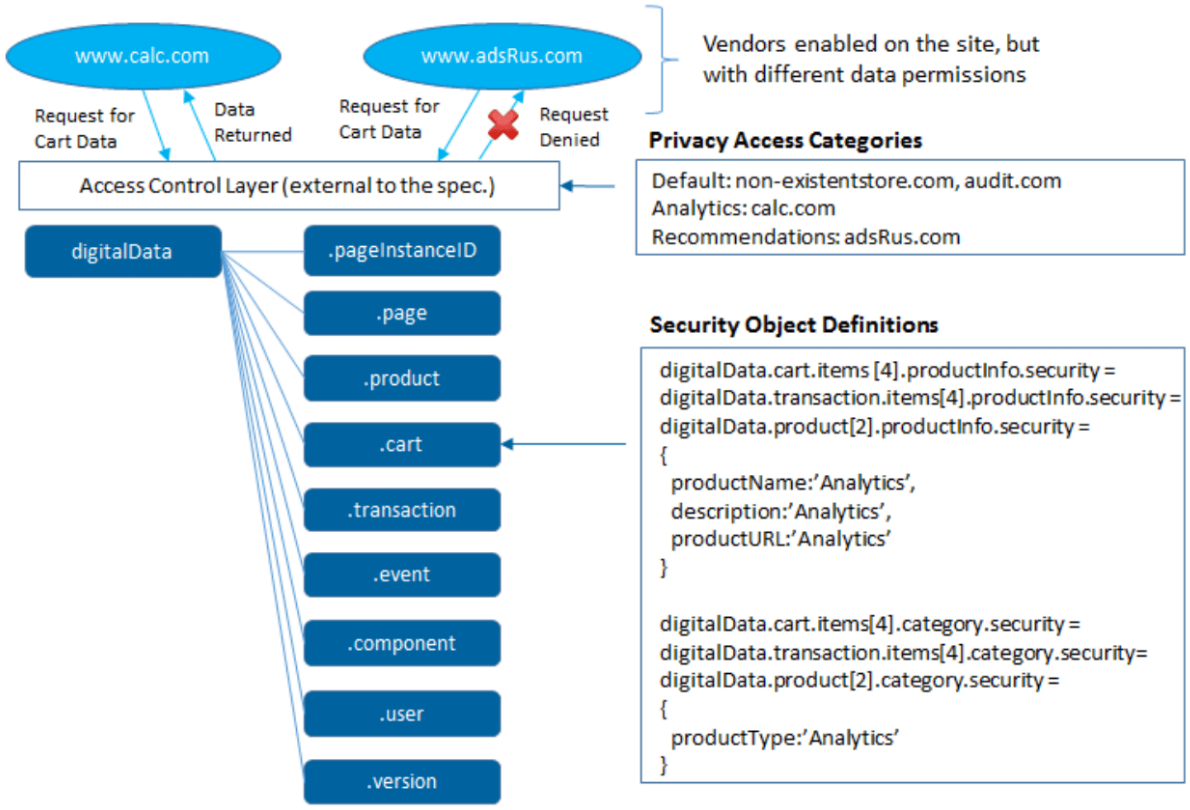

Figure 2: Enforcing data access control using privacy and security setup.

CEDDL specification allows you to add privacy and security preferences. That said, specification doesn't have any way to capture or enforce these preferences.

To enforce and capture these preferences you may have to rely on vendor provided tools. Many tag management systems like Tealium, BrightTag already provide built-in privacy modules.

Final Thought

CEDDL specification is just starting. At first instance, CEDDL looks very promising and solves a big pain — lack of a data layer standard. For me a major take away was ability to extend the specification with ease for variety of verticals.