WHITEPAPER

Data Layers and Tag Management Systems for Technical Audiences

Eduard Boguslavsky
Senior Implementation Consultant II
Blast Analytics

Table of Contents

TAG MANAGEMENT SYSTEM

DATA LAYER

DATA LAYER BEST PRACTICES

Target Audience

This document is intended for technical architects, developers, and similar technical stakeholders, as well as project managers of technical projects. This document doesn't contain any specific action items or business requirements. Instead, it provides an explanation of what analysts and marketers use tag management systems (TMS) for, and how these systems interact with the data layer you'll build. Additionally, it provides developer-oriented recommendations for how to approach architecting a data layer in a language and platform-agnostic way.

Tag Management System

WHAT'S A TAG MANAGEMENT SYSTEM?

A tag management system (TMS) is **software that's embedded on websites, and sometimes apps, to provide users an interface for injecting third-party code** (usually in the form of images, iFrames, and JavaScript files). These systems often have multiple, frequent users and may include third-parties (such as vendors and consultants). Typical users of the TMS are marketers and analysts, but may also include web developers, product owners, and consultants. For this reason, extending your organization engineering governance within the platform is important to consider.

The most common tag management systems are **Google Tag Manager, Adobe Launch** (which has replaced Adobe Dynamic Tag Manager), and **Tealium iQ.**

TEALIUM

Google Tag Manager

Adobe

Tag management systems allow “on-the-fly” customization of marketing, analytics, and functional third-party tools. Without a TMS, developers would be asked constantly to make small modifications to the site.

HOW ARE TAG MANAGEMENT SYSTEMS USED?

Tag management systems solve a unique problem in web applications in that they abstract the needs of marketers and analysts from the development of the application. Instead of having business stakeholders asking developers to modify code in order to trigger events and send data to different third-party tools, utilizing a TMS empowers those users to create their own business logic.

Users of tag management systems depend on information in the **data layer** to create rules for firing their code, as well as for the data they send to marketing and analytics platforms.

Tag management systems are necessary because the business requirements of marketing and analytics stakeholders are often urgent and can change quickly. Tag management systems allow “on-the-fly” customization of marketing, analytics, and functional third-party tools. Without a TMS, developers would be asked constantly to make small modifications to the site. **Management of the various tags is much more efficient in a TMS** than being hard-coded into the website or building custom solutions. Furthermore, the central point of management of analytics and marketing tags makes it an ideal solution to integrate privacy consent management and more easily adhere to an organization’s data governance objectives.

HOW CAN MY TEAM BEST SUPPORT THE TMS IMPLEMENTATION?

In short, by **collaborating, communicating, and knowledge sharing** around:

- The design and implementation of a data layer
- Specifics of the organization as it relates to the implementation

It's in the **interest of developers and other adjacent parties to participate** in the creation of the data layer and the governance of the TMS. Because a TMS can inject any code into the page, it can involve many risks. Injected code:

- Can interfere with site functionality
- May not meet accessibility or browser requirements
- May impact site performance
- May cause issues that impact the development team, such as javascript errors, and such problems may initially be brought to the developers for diagnosis

Technical teams can leverage the TMS and analytics platform for telemetry and error reporting if an organization hasn't already invested in tools specific to this purpose.

These **risks are mitigated** by:

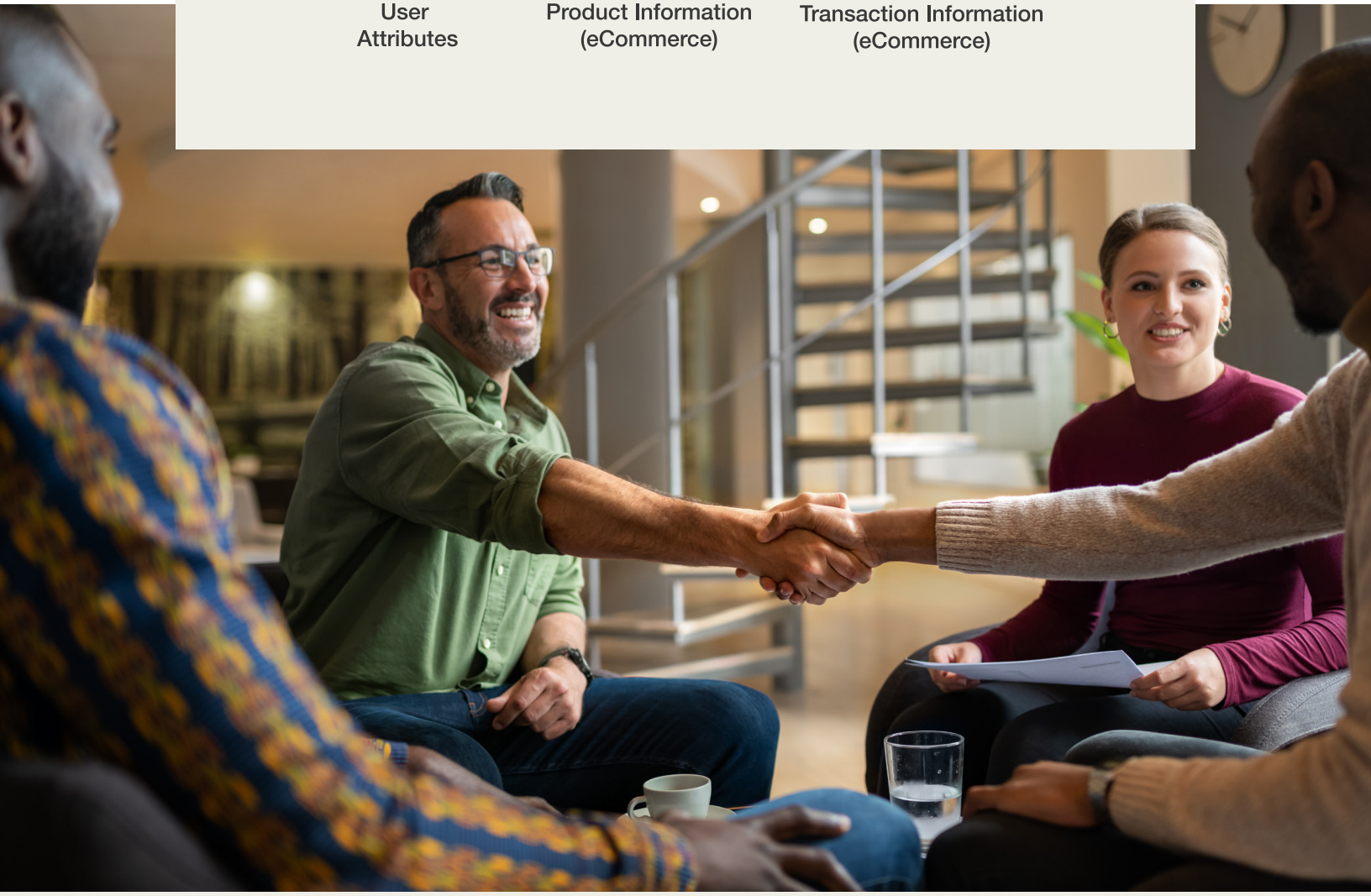
- Communication between developers and those involved in the TMS implementation
- Robust data layer implementation that removes the need for TMS users to implement risky code and triggers, such as timeouts, mutation observers, and the use of Document Object Model (DOM) scraping
- Coordinating around governance of the TMS, especially with custom code

An **additional benefit of collaboration** is that technical teams can leverage the TMS and analytics platform for telemetry and error reporting if an organization hasn't already invested in tools specific to this purpose.

WHAT TYPES OF SYSTEMS DOES THE TMS CONNECT TO?

There are two primary types of connections made by tag management systems: analytics and marketing/advertising. There's considerable overlap between the needs of these two tag types, and typically most needs are covered by the analytics platform. Some typical types of information sent to the TMS include:

- Taxonomic Information
- Conversion Events
- Telemetry Variables
- Page and Content Information
- User Attributes
- Product Information (eCommerce)
- Transaction Information (eCommerce)



Data Layer

WHAT'S A DATA LAYER?

A data layer is typically a javascript object or collection of variables that provide specific information to the TMS for routing to other platforms. There's no singular way to create a data layer; it's an art and is as opinionated as any other decision in the design and engineering of an application.

A specification called **Customer Experience Digital Data Layer (CEDDL)** was conceived to be a standard in 2013. Today, it's uncommon to see it implemented, **and while Blast does not recommend relying on it to architect the data layer,** it's useful to consult when thinking about the kinds of fields to include and the taxonomy of a data layer. For those who are already using Google Analytics (GA) and Google Tag Manager (GTM), **Google's GTM documentation** is the preferred place to start.

From the specification, we recommend studying the high-level object structure and thinking of three key types of variables: page, user, and event. (Ecommerce implementations will have additional ones, covered in CEDDL, **GTM's Enhanced Ecommerce (EE) guide**, and **Tealium's retail data layer definition**.)

We recommend studying the high-level object structure and thinking of three key types of variables: page, user, and event.

Page attributes describe the page beyond DOM and meta information. It's used to incorporate your organization's content strategy and taxonomy. User attributes describe known user characteristics; they're used for remarketing, testing, and personalization. **Event tracking usually makes up the bulk of a data layer implementation:** these are primarily used to record user interactions for analytics tracking, testing, personalization, and some marketing platforms. Events can also be used to feed delayed information (such as from an API call) or indicate that something not controlled by the user has occurred (e.g., a modal popup, A/B test variant, etc.).

By implementing a thorough data layer, you minimize business user frustration and urgent requests for your development team.

WHY DO I NEED TO BUILD A DATA LAYER?

A data layer provides a **single, canonical source of data that's mapped across analytics and marketing systems.** By implementing a thorough data layer, you minimize business user frustration and urgent requests for your development team. The lack of a data layer will require business users to create complicated rules that are subject to inconsistent or volatile elements in your code (increasing risk).

For instance, users may write logic that depends on page titles or CSS selectors, which may change between pages or releases because it wasn't intended to support such use. These users might instead request urgent ad-hoc changes to help them trigger their rules or set the correct data. **A thorough data layer implementation anticipates the vast majority of such needs** and should mean your team doesn't field requests except in very specialized cases or when new features are developed.

HOW DOES THE RIGHT DATA GET INTO THE TMS?

Data that's used in your TMS should come from your data layer. In some cases, variables in the data layer are a replication of variables that exist elsewhere through your other platforms (e.g., your CMS). Routing this information into the data layer is easy, and we can provide specific instructions on the format it should take. Sometimes, this data will require some manipulation, such as casing.

Though the TMS has the ability to crawl the DOM to grab values, that negates a **primary reason to use a data layer: to decouple the data feeding analytics and marketing systems from the DOM.** By decoupling the data layer and DOM, consistency and data hygiene are assured. DOM scraping depends on CSS selectors, hrefs, and text values, which are volatile.

Determining what values to send into the TMS is usually **decided by a thorough but efficient investment of time by all relevant parties.** Business and marketing stakeholders will describe their marketing platforms and business questions. Engineering teams should do a walkthrough of the relevant technologies (e.g., content management system, ecommerce engine). Then analysts, strategists, and implementers can understand what valuable information exists, as well as its relative consistency and effort to expose (both in terms of engineering level of effort and performance implications, if any).

By decoupling the data layer and DOM, consistency and data hygiene are assured. DOM scraping depends on CSS selectors, hrefs, and text values, which are volatile.

Additionally, those implementing the TMS may work with engineering teams on data layer implementation best practices and help identify metrics that are valuable for telemetry and enable reporting within the analytics tool.

Data Layer Best Practices

RECOMMENDATIONS

How you implement the data layer is very dependent on your marketing technology stack, business and marketing needs, and the available resources. That said, **we strongly believe it's valuable to develop an abstract architecture.** This may mimic other paradigms you follow. For example, if your CMS is headless, you may build the data layer by building it out of the data sent to the front end. The goal is to minimize ad-hoc work that needs to be done on particular pages to fire off necessary events, and instead connect it to page templates or components.

While you're not dealing with developers, some of your stakeholders are likely technical and can benefit from a walkthrough of your architecture.

It's important to architect your data layer in such a way that it'll be easy to support changes as business requirements, application logic, and features evolve.

To be successful, we recommend keeping the following best practices in mind:

- **Share** as much as you can about how your application works; while you're not dealing with developers, some of your stakeholders are likely technical and can benefit from a walkthrough of your architecture and discover valuable variables that can be easily incorporated in the data layer.
- **Suggest** features to track and participate in conversation about measurement plans. Your team can leverage tracking to answer questions without additional investment or novel solutions.
- **Request** basic telemetry and reporting; as above, you can get important statistics without much additional effort.
- **Listen** to understand the needs and capabilities of analytics, advertising, and marketing tools.
- **Involve** analytics and marketing stakeholders early in feature and product development so they can provide data collection requirements with plenty of lead time.
- **Include** analytics teams in your Program Increment (PI) planning or similar high-level planning conversations to raise awareness and prevent urgent ad-hoc requests close to launch.

Suggested Reading

The Data Layer
Author: Simo Ahava

Developer Guide: Using a Data Layer
Author: Google Tag Manager

What is a Data Layer?
Author: Tealium

Understanding the Data Layer
Author: IPullRank

Customer Experience Digital Data Layer
Author: W3C
(Warning: This document is outdated and should only be used to generate ideas for the content of your data layer)

The Event-Driven Data Layer
Author: Jim Gordon

Roseville HQ
950 Reserve Dr., #150
Roseville, CA 95678

San Francisco
156 2nd St.
San Francisco, CA 94105

New York City
54 West 40th St.
New York, NY 10018

Seattle
500 Yale Ave. North
Seattle, WA 98109

Los Angeles
1601 Vine St.
Los Angeles, CA 90028

Chicago
220 North Green St.
Chicago, IL 60607

Dallas
1920 McKinney Ave.
Dallas, TX 75201

Washington. D.C.
1440 G St. NW
Washington, D.C., 20005

London
22 Upper Ground
London, UK SE1 9PD

www.blastanalytics.com | solutions@blastam.com | 888.252.7866