utag.js Order of Operation

Version 1.4

How utag.js works

The sequence of events from loading to sending and how they impact your Tealium iQ implementation
Table of Contents

Introduction

   About Tealium, Inc. ..............2

   Purpose of this Document ........2

Order of Operation Overview

   Web Page Processing Order.........3

   Order of Operation ...............3

   Processing of utag.js Details .......4

Appendix

   Data Source Mappings .............6

   DOM ................6

   Extension Scope .................6

   Tag Bundling ..................6

   Wait Flag ....................6

   utag.view() ..................6
Introduction

About Tealium, Inc.

Founded in 2008, Tealium is the leader in enterprise tag management, serving some of the most demanding clients worldwide. Tealium’s vendor-agnostic solution makes it easy for organizations to add, modify, and remove all of their digital marketing page tags through a single line of code.

The company differentiates itself through an intuitive web-based console that allows business users to manage all of their web and mobile tagging requirements without JavaScript knowledge. Tealium offers turnkey integration with 600+ online vendors, superior scale and performance, and a deep feature set to help organizations increase marketing agility, reduce tagging costs and improve site performance. Tealium enables its customers to leverage the entire digital marketing ecosystem.

Purpose of this Document

This document is designed to familiarize you with the order of operations for Tealium’s utag.js script.
Order of Operation Overview

Web Page Processing Order

To understand how Tealium’s Tag Management System (TMS) loads, we should look at how a browser loads a web page.

1. First, the browser requests the web page’s HTML (Hyper Text Markup Language) document.

2. The browser begins to download and parse the HTML document starting at the top-left and continuing down to the bottom-right. As the browser reads through the HTML it encounters various elements that make up the content of the web page. These elements can be any images, links, text, or scripts like the Tealium script (utag.js).

3. The browser continues to download these elements as it encounters them while it downloads the HTML document itself. In the case of regular scripts, the browser must stop downloading the HTML and interpret the script before continuing on to the rest of the HTML. Smart scripts, like utag.js, have a built-in halt that pauses the script and allows the browser to continue downloading the HTML. The halt is released when the browser signals that the HTML has been fully downloaded and parsed into the Document Object Model (DOM). This signal is typically called ‘DOM Ready’ (more on this later).

4. The browser begins to render the web page on the visitor’s screen.

NOTE: DOM Ready is a signal from the browser that it has downloaded and fully parsed the web page’s HTML into the DOM. With the DOM completed, all of the web page’s elements are available to interact with scripts like utag.js. Waiting for DOM Ready before performing any operations ensures that the data required by utag.js for your various third-party Tags are available. Moreover, waiting for DOM Ready also means that utag.js is not getting in the way of the web page loading for your visitor.

Order of Operation

The utag.js operations flow in a very specific order and understanding that order can prevent errors in your implementation. The default order of operations is:

1. Run Extensions scoped to “Pre-Loader”

2. Process Data Layer

3. Evaluate Load Rules

4. Run Extensions scoped to “All Tags”

5. Load Bundled Vendor Tags and Vendor Tags with their Wait Flag set to “Off”

6. Load Vendor Tags, their Extensions, and Mappings

7. DOM Ready, DOM Ready scoped Extensions, and the Send Function

8. After Page Load: utag.view() and Advanced Configurations
Processing of utag.js Details

Step 1: Pre-Loader Extensions

Pre-Loader extensions run first. Since Pre-Loader extensions run before everything else, it is vital that they not rely on any of the Tealium elements (data layer, other extensions, etc.) that run later. Attempting to do so could result in the page loading improperly, or not at all. The order in which Pre-Loader extensions are processed is based upon their order in the Tealium iQ console. The Tealium Privacy cookie is set at this point, if it does not yet exist.

Step 2: The Data Layer is Built

The utag_data object, also referred to as the Universal Data Object (UDO), passes data from your web page to Tealium iQ. The variables and data contained in utag_data are combined with data from the rest of the web page, 1st-party cookies, and the DOM to create the utag.data object. While the utag.data object contains all the web page’s data, only the data you identify as data sources are used in the Tealium iQ console for mappings.

Step 3: Load Rules

The Load Rules are evaluated. The Load Rules determine if particular Tags load on any given web page. If a Load Rule determines that a Tag should not load on a web page, that Tag and any extensions scoped to it do not load. At this point, all of the data on the page may not be available as the browser has not yet finished parsing. The Load Rules are re-evaluated later, when the rest of the page’s data is available, to make certain they evaluate correctly.

Step 4: Extensions Scoped to “All Tags”

Any extensions scoped to “All Tags” run. These extensions execute in the order in which they appear in Tealium iQ. This scope indicates that the data that come from these extensions are available to all the Tags in your profile.
Step 5: Load and Fire Vendor Tags with Wait Flag set to “Off” and Fire Bundled Vendor Tags

Two types of Tags fire at this point: bundled Tags and Tags with their Wait Flag set to “Off”. Bundled Tags have their code included directly within utag.js. The Wait Flag is an advanced setting that controls if the tag should fire at the DOM-ready event (default) or immediately as utag.js loads.

These Tags will fire based on the initial evaluation of Load Rules.

Step 6: Vendor Tags, their Extensions, and Mappings

Depending on whether or not the Load Rules permit, the vendor Tag libraries load next. If any of those Tags have extensions scoped to them individually, those extensions load along with the Tag. After the Tag libraries and Tag-specific extensions are loaded, the page data variables you identified as data sources are mapped to the vendor Tags’ destination variables. As discussed earlier, even though the Tag libraries load, they don’t do anything until the DOM Ready signal is received.

Step 7: DOM Ready and the Send Function

When the browser sends the DOM Ready signal, DOM Ready-scoped extensions and Tag-specific extensions execute and data source mappings are processed. The Send function collects the final vendor Tag requests together and sends them to each of the vendor services. The order in which you have placed your Tags in Tealium iQ is the order in which the data is sent to each vendor.

Step 8: After Page Load: utag.view() and Advanced Configurations

If you use the utag.view() function to track views for page content dynamically loaded after the page has finished loading, load rules are re-evaluated. This is true only for utag.js versions 4.26 and later. Any versions of utag.js prior to 4.26 do not re-evaluate load rules for utag.view().

Additionally, the utag.js file has a configuration (utag.cfg.load_rules_at_wait) that will re-evaluate load rules at DOM Ready when set to true. This is an advanced customization and we recommend that you contact your Account Manager before implementing it. This configuration is only available with utag.js versions 4.26 and later.
Appendix

Data Source Mappings

When you map a data source to a destination, you create a connection between them. Changing the value of a data source propagates the change to the corresponding destination variables. This becomes important in Tag Management because data source mapping is what allows you to send your pages’ variables to numerous third-party Tags through Tealium iQ. Mapping is not restricted to a one-to-one relationship; you can map a single data source to multiple third-party Tag variables.

DOM

DOM is an acronym for Document Object Model. What is the Document Object Model? Simply put, the DOM is the structure of a web page and provides a way for a scripting/programming language, like JavaScript, to interact with a web page or document and all of its elements.

Extension Scope

What do we mean by scope? Scope indicates when an extension executes during the processing of utag.js. Extensions can be scoped to a specific vendor Tag, such as Google Analytics or SiteCatalyst, or scoped to a specific event such as DOM Ready.

Tag Bundling

Tag Bundling is a process that allows you to combine all of your Tag requests into a single large request, rather than making smaller requests for each Tag.

Wait Flag

The Wait Flag is an Advanced Setting that determines whether or not a Tag waits for the DOM Ready signal before it executes. By default, this Flag is set to ‘Yes’. Setting this Flag to ‘No’ causes the Tag to begin loading before the DOM Ready signal is sent. The various elements the Tag relies on may not yet be available on the page if the browser has not finished parsing the DOM, so loading the Tag before the DOM Ready signal could result in the Tag loading improperly or not at all.

utag.view()

Tealum uses the utag.view() function to track page view events. When the utag.js file loads on a page, it automatically calls this function to track the initial page load as a page view event. Do not hard code a utag.view() call into your page’s source code, as it will send a second page view event, in addition to the one sent by the utag.js file, and distort your reports.
UTAG.JS ORDER OF OPERATIONS (Simplified)

**KEY**

- Universal Data Object (UDO)
- Extension (Default)
- Extension (Alternate)
- Load Rules
- Vendor Tag
- In-Scope UDO Reference
- Inactive Object or Operation

**UDO Reference Variable Scope**

**GLOBAL SCOPE**

- `utag_data`
  - Pre Loader

**LOCAL SCOPE**

- `b`
  - All Tags
  - Tag-Specific

- `d`
  - Load Rules

---

© 2016 Tealium, Inc.
utag Waterfall Non Ready Wait

---

Page Load
  utag Load
    Pre Loaders
      Initialization
        utagMaker
          over ride flags set
            DOM Ready
              Extension Handler Defined
                Pre-load

Set-up Data
  utag_data
    JS Page Var Read
      Read:
        op / op / meta / cookie
        / domain / URL
    Set utag_main
      cookie
    Load Rules
      Processed
    <Det extensions run>
      Load Rules
      Processed
      int
        IF NOT trigger
          Run "All Tag Scope" Extensions
        else
          Load non-ready wait tags
            Create DOM
              ready handler

DOM Time
  Event
    Loading
      Loaded
        Complete
          (dom_complete flag)

DOM Load
  utag_x.js Load
    var setup
    utag LOAD called
      if NOT trigger
        Check for any waiting requests
          Call tag send with any pending requests
        if all tags loaded
          End event
            Set base vars
              if NOT trigger
                <Det extensions run>
            <Det set to true
              trigger any events in ready queue

view/fire event called
  Trigger
    if trigger - tags not loaded
      post event onto tracker queue
    set-up Data
      base vars merged into:
        read:
          op / op / meta / cookie
          / domain / URL
    IF - Fire specific tag
      Run "All Tag Scope" Extensions
        Tag X loaded
          Yes
            Fire Send
            Load tag
          No
    IF - other load rules
      <Det extensions run>
        merge event data with utag_data
        Load Rules
        Processed
        Run "All Tag Scope" Extensions
        Tag X loaded
          Yes
            Fire Send
            Load tag
          No
        else
          trigger send for load event tags
            <Det extensions run>
              update utag_main
              cookie
            call back called

---

TEALIUM, INC ©2016
Order of Operation
utag Waterfall with Ready Wait

Order of Operation