TEALIUM WEB PLATFORM INTEGRATIONS

Tealium Adobe Experience Manager (AEM) Integration

Tealium Plugin for Adobe AEM and CQ Install Guide

2 May 2016

Table of Contents

Overview

This document describes how to install, configure and use the **Tealium Tag Management Plugin** for **Adobe CQ** / **Adobe Experience Manager**. This plugin integration provides a framework for building the utag_data JavaScript object in the HTML source. The utag_data object contains the page-specific Data Layer of key/value pairs.

Description

This plugin allows integration of Adobe CQ / Adobe Experience Manager instance with Tealium Tag Management (Tealium iQ). The plugin contains two versions, one for Adobe CQ and one for Adobe Experience Manager. Each version includes two components that enable the data layer for Tealium Tag Management.

Tealium Component works only with server-side data. The data can be parsed from the request attributes, request parameters, and page properties. However, the client-side data obtained from JavaScript will not be parsed.

Tealium Asynchronous JS Component works with both server-side and client-side data. This component can handle asynchronous requests.

Configuration Page contains a list of variables for sending to Tealium as well as provides UI to specify Tealium settings (account, profile, environment, etc).

Compatibility

This Integration is currently compatible with Adobe CQ / Adobe Experience Manager version 5.5 and higher.

Installation

The following describes the steps to integrate the plugin with Adobe CQ / Adobe Experience Manager as well as validate that everything is working correctly. Note that all links below reference the default Adobe CQ / Adobe Experience Manager install path – http://localhost:4502.

Tealium

Steps to Configure Tealium

1. Log in to Tealium iQ.
2. Create a new tag (for instance, Facebook Like Button).
3. Choose **"Facebook"** Div ID to render your tag.
4. Choose **"All pages"** as a load rule.
5. You can add additional tags with different load rules for additional testing

Adobe CQ / Adobe Experience Manager

Steps to Configure Adobe CQ / Adobe Experience Manager

1. Install either **"Tealium/integration-adobe-cq"** or **"Tealium/integration-adobe-aem"** package from GitHub location
    a. https://github.com/Tealium/integration-adobe-cq
    b. https://github.com/Tealium/integration-adobe-aem
2. Install **"tealium-examples"** package. The "tealium-examples" directory is in the repository downloaded from the first step.
3. Open the configuration page using the following link – http://localhost:4502/cf#/content/tealium-plugin/configuration-page.html.
4. *Note: Touch ID is not supported for the configuration page.*
5. Open the global configuration component dialog (this component is the first one on the page).
6. Check the **"Enable"** checkbox to enable the plugin.
7. Enter your Tealium account information in the following fields: **"Account"**, **"Profile"** and **"Environment"**. Set Environment value to "dev" for testing locally in your sandbox.
8. Save your changes.

9. For Adobe CQ, open the following link
   http://localhost:4502/content/tealium-examples/cq/geometrixx-outdoors/en.html
10. For Adobe Experience Manager, open the following link
    http://localhost:4502/content/tealium-examples/aem/geometrixx-outdoors/en.html
11. Verify that the Facebook like button is displayed on the opened page.
12. *Troubleshooting: Remove the **"cf#"** from your URL when testing your integration on an author instance*

Configuration

The plugin allows configuring global and page type settings where you can enable or completely disable the plugin, specify your Tealium account details, and configure what Data Layer values should be displayed on each specific page. The final set of Universal Data Object (UDO) properties will contain all properties defined in the both Global Configuration and Page Type Configuration of that specific page template.

The application might have more than one **Tealium Configuration Page**, but only one will be used. The default configuration is placed under **"/content/tealium-plugin/configuration-page"** and it can be opened by the following link – http://localhost:4502/content/tealium-plugin/configuration-page.html.

Global Configuration

The global configuration contains the following settings that affect the whole application.

Global Settings

Setting	Description
Enable	Enable or disable the plugin for the site
Account	The **Tealium Account** for connecting with Tealium
Profile	The **Tealium Profile** for connecting with Tealium
Environment	The **Tealium Environment** for your publish (i.e. "dev", "qa", or "prod")
Enable utag.sync.js	Enable or disable including the "utag.sync.js" script in the page
Enable Custom UDO	Enable or disable rendering the data layer object (utag_data) in the page

Global Universal Data Object (UDO) Variables

Setting	Description
UDO Name	The name of a data layer event attribute (i.e. “page_name”)
UDO Value	The value of the data layer event attribute (i.e. “home”)
Render as Array	Set if the event attribute value should be rendered as a JavaScript Array
Is Dynamic Value	**Check** if value should be replaced on the server-side with a page property, request parameter, or JavaScript variable passed to “TealiumAsyncJSManager” **Uncheck** if value should be rendered as is (static)

Page Type Configuration

The page type configuration allows configuring UDO variables for a specific page by specifying a path of that page template as well as a list of UDO properties to be rendered on all page instances of that page template only. Most likely you will need to use more than one page type configuration. They can be included in a page parsys using sidekick.

Page-Specific Settings

Setting	Description
Page Template Path	The path to a page template; if the “sling:resourceType” property of a requested page matches what you have specified in “Page Template Path”, this configuration of the Custom UDO properties will be applied to the page
Enable Custom UDO	Enable or disable rendering the data layer object (utag_data) in the page
Page-Specific UDO Variables	A list of properties to be rendered inside the data layer object (utag_data) on each page of that page template; contains the same fields as in the Global Configuration: “UDO Name”, “UDO Value”, “Render as Array”, “Is Dynamic Value”

Components

The plugin contains two components: **Tealium Component** and **Tealium Asynchronous JS Component**. Both the components are written in JSP and Sightly, and are ready to be included into other components or pages. However, each page should contain only one component.

Tealium Component is rendered on a page as soon as it is included. This component works with server-side variables and page properties and does not work with JS variables.

Tealium Asynchronous JS Component is used to handle asynchronous requests. All Tealium-related code will be added to a page when all registered JS methods (actions) are executed. This component works with server-side variables, page properties and JS variables.

Tealium Component

This component is used to include Tealium-related code (the **“utag_data”** object variable and all necessary JavaScript) in a page as soon as the component is rendered. This component should be used when there are no asynchronous JS requests and should executed before sending data to Tealium, or no JS variables should be passed to Tealium.

This component renders two pieces of JS code: the **“utag_data”** JS variable and the **“utag.sync.js”** script (if enabled on the configuration page).

Note: All server-side variables should be set before rendering the component.

Adobe CQ

There are two ways of how to include the component in a page.

1. Include the component using the **“<cq:include>”** tag (preferred method)

```
<cq:include path="tealium" resourceType="tealium-plugin/components/tealium-plugin"/>
```

2. Include the component in a parsys using sidekick

Adobe Experience Manager

There are two ways of how to include the component in a page.

1. Include the component using the **“data-sly-resource”** statement (preferred method)

```
<div data-sly-resource="${ 'tealium' @
resourceType='/apps/sightly-tealium-plugin/components/tealium-plugin' }" />
```

2. Include the component in a parsys using sidekick

Component Variable Scopes

The component uses the following order in order to replace dynamic values on the server-side.

1. **Request Attribute Scope**
    - Using the standard **“<c:set>”** JSTL tag. For example, in order to set value **“777”** to the **“productID”** variable

      ```
      <c:set var="productID" value="777" scope="request" />
      ```

    - Using the Sightly JS API. For example, in order to set value **“777”** for the **“productID”** variable

      ```
      <div
      data-sly-use="${'/apps/sightly-tealium-plugin/components/tealium-plugin/setAtt
      ribute.js' @productID='777'}" />
      ```

2. **Request Parameter Scope**
    - The variable should be set in the request path

3. **Page Property**
    - The variable should be defined in properties of the requested page

Tealium Asynchronous JS Component

This component handles asynchronous requests and can work with both server-side and client-side data.
Note: All server-side variables should be set before rendering the component.

Adobe CQ

In order to include the component in a page, use the following code

```
<cq:include path="tealium-async-js-plugin"
resourceType="/apps/tealium-plugin/components/tealium-async-js-plugin"/>
```

Adobe Experience Manager

In order to include the component in a page, use the following code

```
<sightly data-sly-resource="${@path='async-plugin-launcher',
resourceType='/apps/tealium-example/aem/components/async-plugin-launcher'}"
data-sly-unwrap></sightly>
```

Component Variable Scopes

The component uses the following order in order to replace dynamic values on the server-side

1. **Request Attribute Scope**
    - The standard **“<c:set>”** JSTL tag. For example, in order to set value **“777”** to the **“productID”** variable

```
<c:set var="productID" value="777" scope="request" />
```

- The Sightly JS API. For example, in order to set value **“777”** for the **“productID”** variable

```
<div
data-sly-use="${'/apps/sightly-tealium-plugin/components/tealium-plugin/setAtt
ribute.js' @productID='777'}" />
```

2. **Request Parameter Scope**
    - The variable should be set in the request path

3. **Page Property**
    - The variable should be defined in properties of the requested page

4. **JavaScript Variable**
    - The variable should be added to **“TealiumAsyncJSManager”**

OSGi Components

The plugin contains two Servlets and one Service. The OSGi Components are used by the Tealium Asynchronous JS Component and by the Configuration Page.

OSGi Servlets

There are two servlets registered in OSGi.

1. **Tealium Configuration Servlet**
    - The configuration page uses the servlet to initialize in-memory configuration values
    - Path is **“/services/tealium/configuration/init”**
    - HTTP method is GET

2. **Tealium JS Component Servlet**
    - The JS module uses the servlet to merge all server-side and client-side variables with dynamic UDO values to build a final **“utag_data”** JS variable
    - Path is **“/services/tealium/components/js/include”**
    - HTTP method is GET

OSGi Services

The plugin contains one service – **Tealium Configuration Service**. This service configures a path to the configuration page, which should be used by the application. By default, access on a local environment with the following link.

http://localhost:4502/system/console/configMgr/com.tealium.services.TealiumConfigurationService

There is only one configuration field – **“Configuration Page Path”**. In order to change the default configuration page, you just need to enter a path of a new configuration page there (without .html).

JS Module

The plugin contains a JS module – **“TealiumAsyncJSManager”**. This module registers asynchronous JS methods (actions) or just adds JS variables to be dispatched to Tealium together with other UDO variables. The module has the following methods (actions):

Methods (Actions)

Name	Params
registerMethod	`TealiumAsyncJSManager.registerMethod(`**`methodName, elementToListen`**`)` **methodName** - a unique JS method (action) name **elementToListen -** a jQuery element for listening when the method is executed

	When this method is executed, a developer is responsible for triggering the event with the registered **“methodName”** on **“elementToListen”**. `elementToListen.trigger(productDataCaller, data)`
deregisterMethod	`TealiumAsyncJSManager.deregisterMethod(`**`methodName, elementToListen`**`)` **methodName** - a previously set JS method (action) name **elementToListen -** a previously set jQuery element for listening This method should be called if, for instance, a previously registered method was not executed for some reason.
startProcessing	`TealiumAsyncJSManager.startProcessing()` When this method is executed, “TealiumAsyncJSManager” will be ready to render the “utag_data” variable. This method should be executed after all JS methods have been registered.
addValueToResult	`TealiumAsyncJSManager.addValueToResult(`**`data`**`)` **data** - a JS object or JSON to replace a dynamic UDO value This method should be called to replace dynamic UDO values.
renderUtagData	`TealiumAsyncJSManager.renderUtagData()` This method should be called to render the “utag_data” variable in a page.
enableDebug	`TealiumAsyncJSManager.enableDebug()` This method enables displaying debug information on the browser console. By default, this method should be executed at the beginning of a page.