TEALIUM WEB PLATFORM INTEGRATIONS

Tealium Adobe Experience Manager 6.2 Integration

Tealium Plugin 1.1.0 for Adobe AEM 6.2
Install Guide

6 Dec 2016

Table of Contents

Overview

This document describes how to install, configure and use the **Tealium Tag Management Plugin** for **Adobe Experience Manager (AEM)**. 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 Experience Manager instance with Tealium Tag Management (Tealium iQ). The plugin includes two components that enable the Data Layer in a static web page or asynchronous request for 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 Experience Manager version 6.2.

Installation

The following describes the steps to integrate the plugin with Adobe Experience Manager as well as validate that everything is working correctly. All links below reference the default Adobe Experience Manager install path – http://localhost:4502. Note that the geometrixx example (from Adobe) must also be installed for the example.

Tealium

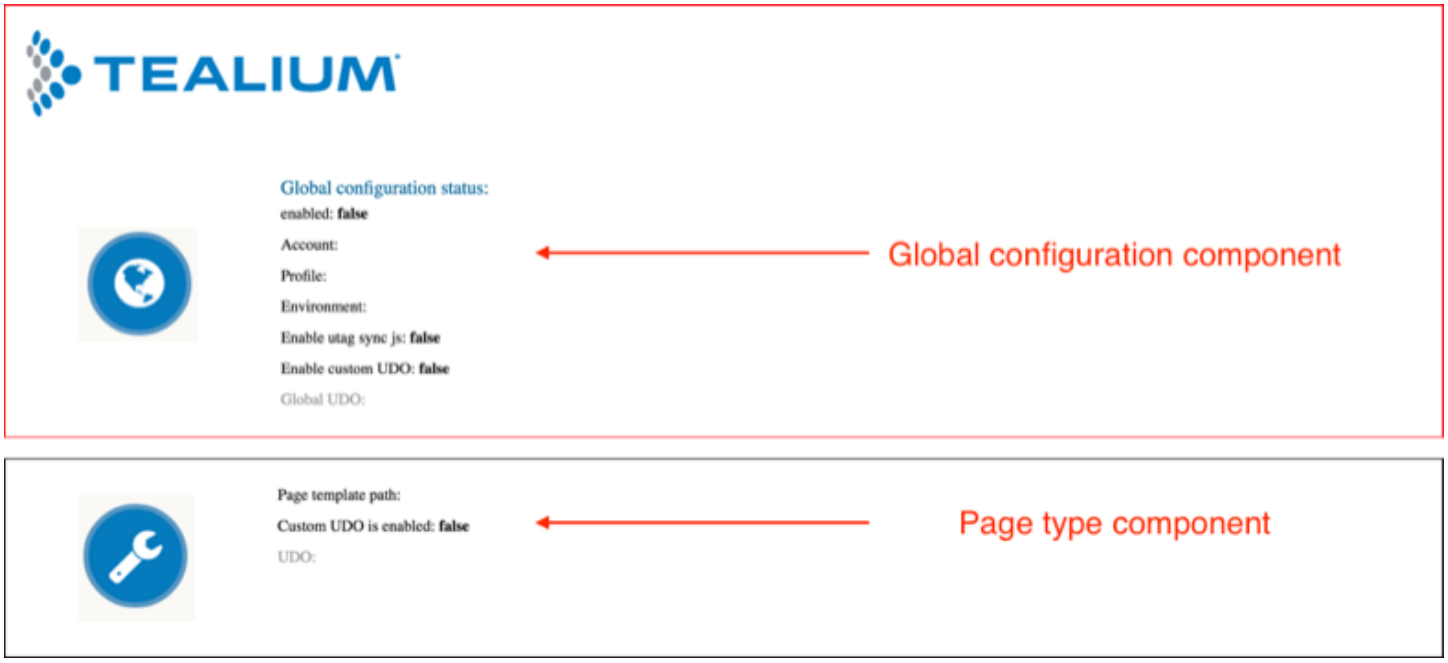
Steps to Configure Tealium

1. Log in to Tealium iQ.
2. Create a new Tag (for instance, Facebook Like Button).
3. For the Tag configuration, enter a Div ID location (i.e. "facebook") to render your Facebook button
4. Choose **“All pages”** as a load rule.
5. You can add additional tags with different load rules for additional testing. For information on Tealium iQ, please visit https://community.tealiumiq.com.

Adobe Experience Manager

Steps to Configure Adobe Experience Manager

1. Login to AEM.
2. Go to CRX Package Manager: http://localhost:4502/crx/packmgr/index.jsp
3. Click on "Upload Package".
4. For more details on AEM, please visit https://docs.adobe.com/docs/en/aem/6-2/deploy.html
5. Install **Tealium/integration-adobe-aem** package from GitHub location
    a. Github "releases" will show a **tealium-aem-plugin-1.1.0.zip** for AEM 6.2.
    https://github.com/Tealium/integration-adobe-aem/releases
6. Install **tealium-aem-examples-1.1.0.zip** as well. This package contains adapted out of the box geometrixx site to test the plugin.
7. Open the configuration page using the following link – http://localhost:4502/cf#/content/tealium-plugin/configuration-page.html?wcmmode=edit.



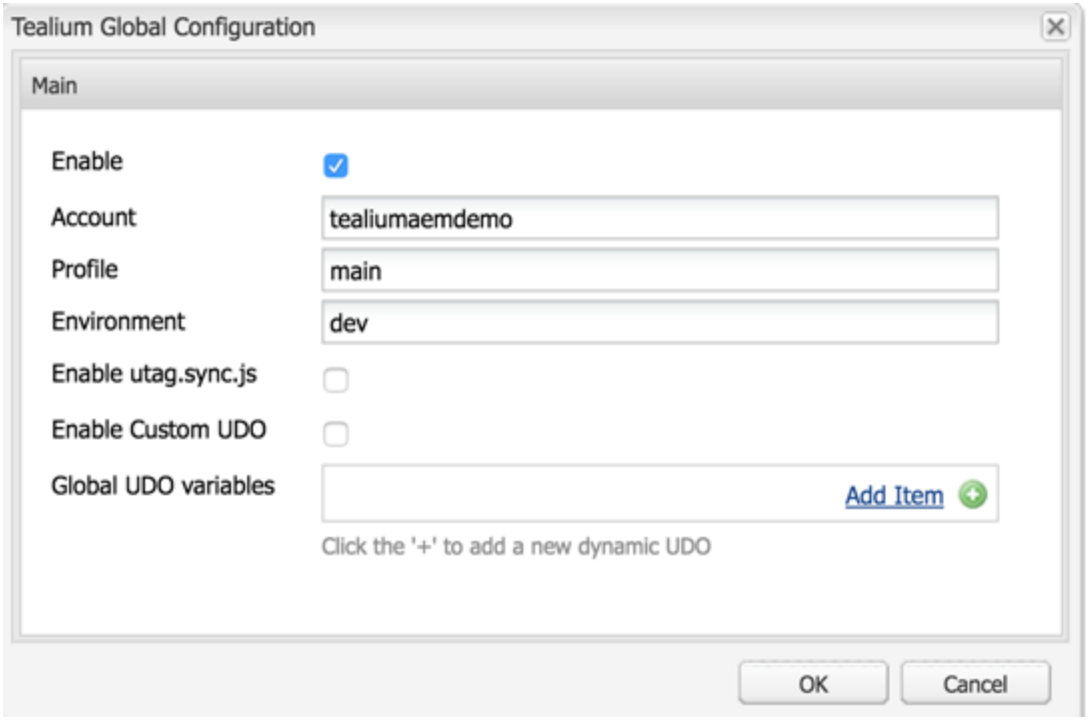
Global configuration status:
enabled: **false**
Account:
Profile:
Environment:
Enable utag sync js: **false**
Enable custom UDO: **false**
Global UDO:

Global configuration component

Page template path:
Custom UDO is enabled: **false**
UDO:

Page type component

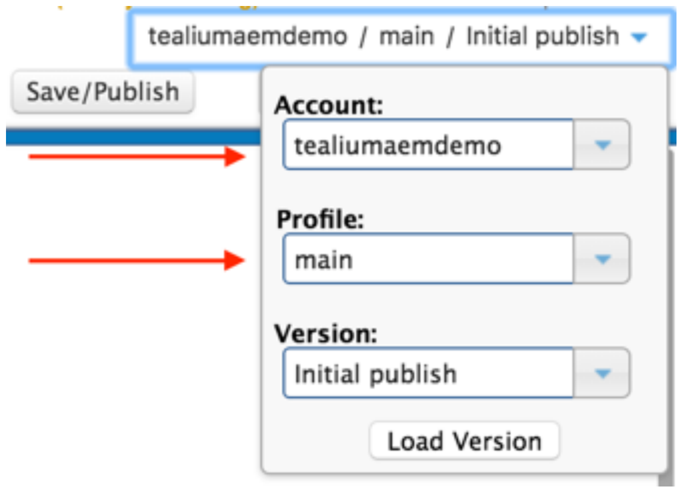
8. Open the Global configuration component dialog by double-clicking on the component or right click and choose "Edit."
9. Check the **"Enable"** checkbox to enable the plugin.
10. Enter your Tealium account information in the following fields: **"Account"**, **"Profile"** and **"Environment"**. Set Environment value to "dev" for testing locally in your sandbox.



Enable	✓
Account	tealiumaemdemo
Profile	main
Environment	dev
Enable utag.sync.js	☐
Enable Custom UDO	☐
Global UDO variables	Add Item

Click the '+' to add a new dynamic UDO

The Tealium Account and Profile values can be found in Tealium iQ in the top-right.



Account: tealiumaemdemo

Profile: main

Version: Initial publish

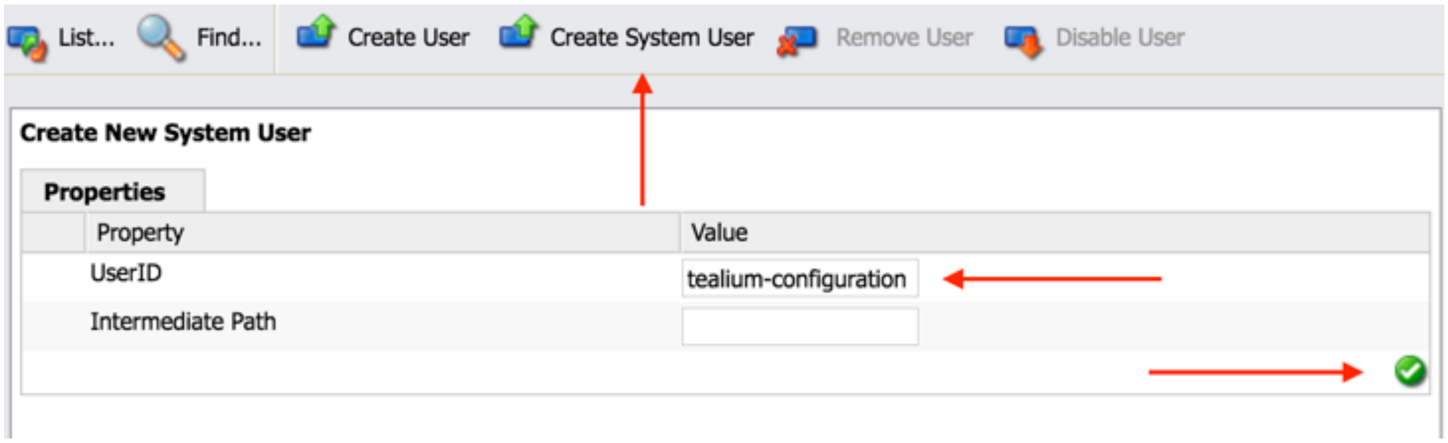
11. Save your changes.

Configure AEM System User Permissions

The plugin bundle requires read-only access to AEM configuration page (by default location in JCR is /content/tealium-plugin/configuration-page). The follow shows steps for creating a new system user, add read-only access to configuration page(s) for the system user, and create service user mapping.

Create System User (Skip this section if you already have a system user)

1. Go to "User Administration" console: http://localhost:4502/crx/explorer/ui/usereditor.jsp?&Path=/home/users
2. Click on "Create System User"
3. Enter System UserID, for instance ‘tealium-configuration’
4. Click on green checkbox



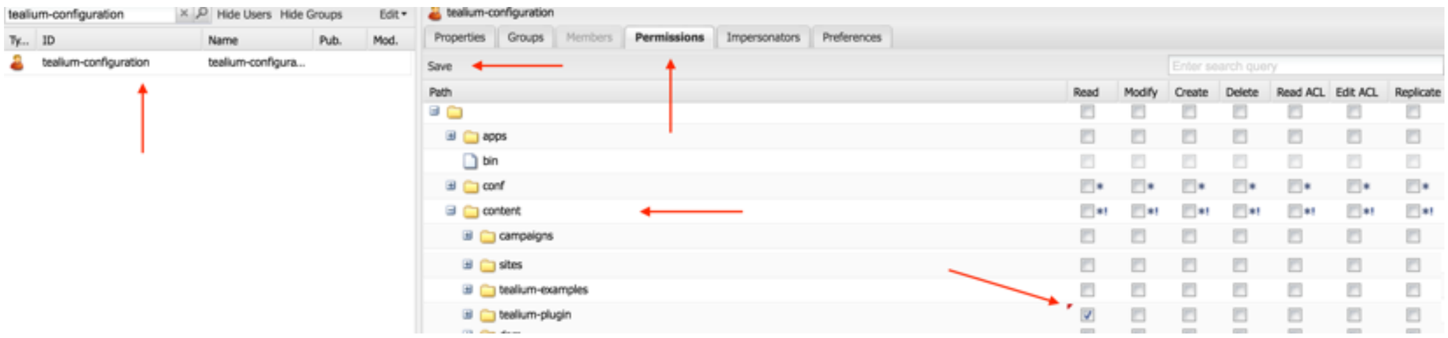
Create New System User

Properties

Property	Value
UserID	tealium-configuration
Intermediate Path	

Configure Read-only Access for Configuration

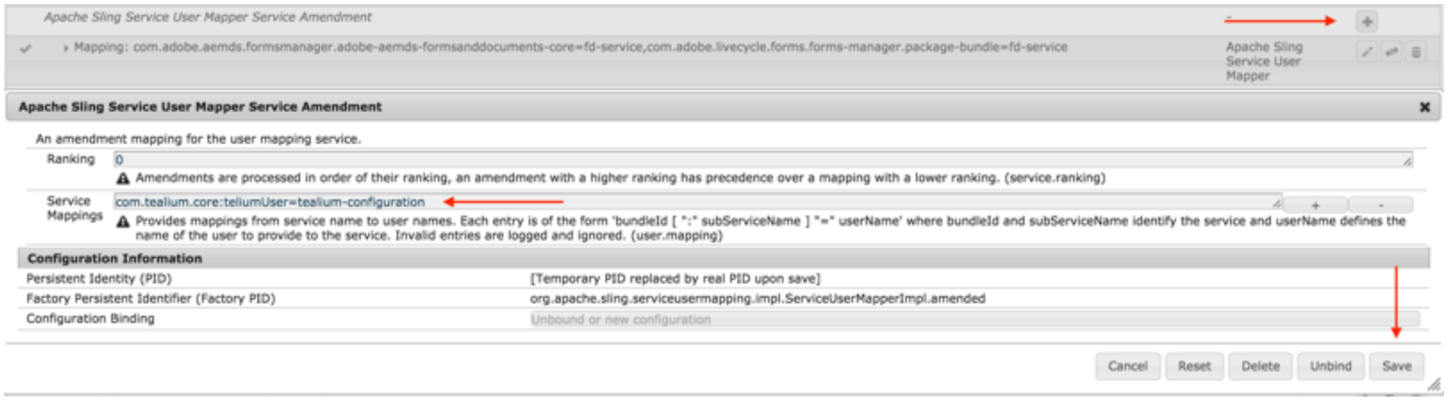
1. Open AEM Security console: http://localhost:4502/useradmin
2. In top left corner search for User that you created (for instance ‘tealium-configuration’)
3. In left result tab double click on found User
4. Open "Permissions" tab
5. In showed hierarchy find content folder
6. Under content node find "tealium-plugin" folder
7. Allow read access for selected user by clicking on checkbox under "Read" section
8. Click on "Save"



Add Mapping for System User to Tealium Plugin

1. Go to Felix Configuration Manager: http://localhost:4502/system/console/configMgr
2. Search for "Apache Sling Service User Mapper Service Amendment"
3. Click on "+" button on right side

4. In "Service Mappings" field add a mapping in format: "[tealium bundle id]:[sub service name]=[system user name]". For the "tealium-configuration" user created earlier, mapping will be "com.tealium.core:tealiumUser=tealium-configuration".


Apache Sling Service User Mapper Service Amendment
Mapping: com.adobe.aemds.formsmanager.adobe-aemds-formsanddocuments-core=fd-service,com.adobe.livecycle.forms.forms-manager.package-bundle=fd-service
Apache Sling Service User Mapper

Apache Sling Service User Mapper Service Amendment

An amendment mapping for the user mapping service.

Ranking: 0

Amendments are processed in order of their ranking, an amendment with a higher ranking has precedence over a mapping with a lower ranking. (service.ranking)

Service Mappings: com.tealium.core:teliumUser=tealium-configuration

Provides mappings from service name to user names. Each entry is of the form 'bundleId [":" subServiceName] "=" userName' where bundleId and subServiceName identify the service and userName defines the name of the user to provide to the service. Invalid entries are logged and ignored. (user.mapping)

Configuration Information

Persistent Identity (PID)	[Temporary PID replaced by real PID upon save]
Factory Persistent Identifier (Factory PID)	org.apache.sling.serviceusermapping.impl.ServiceUserMapperImpl.amended
Configuration Binding	Unbound or new configuration

- Tealium bundle id should be "com.tealium.core"
- The sub service name could be configured via OSGi Services. ‘tealium-configuration’ is default name of the Sub service
- System User could be any System User that has read access to ‘/content/tealium-plugin’ node

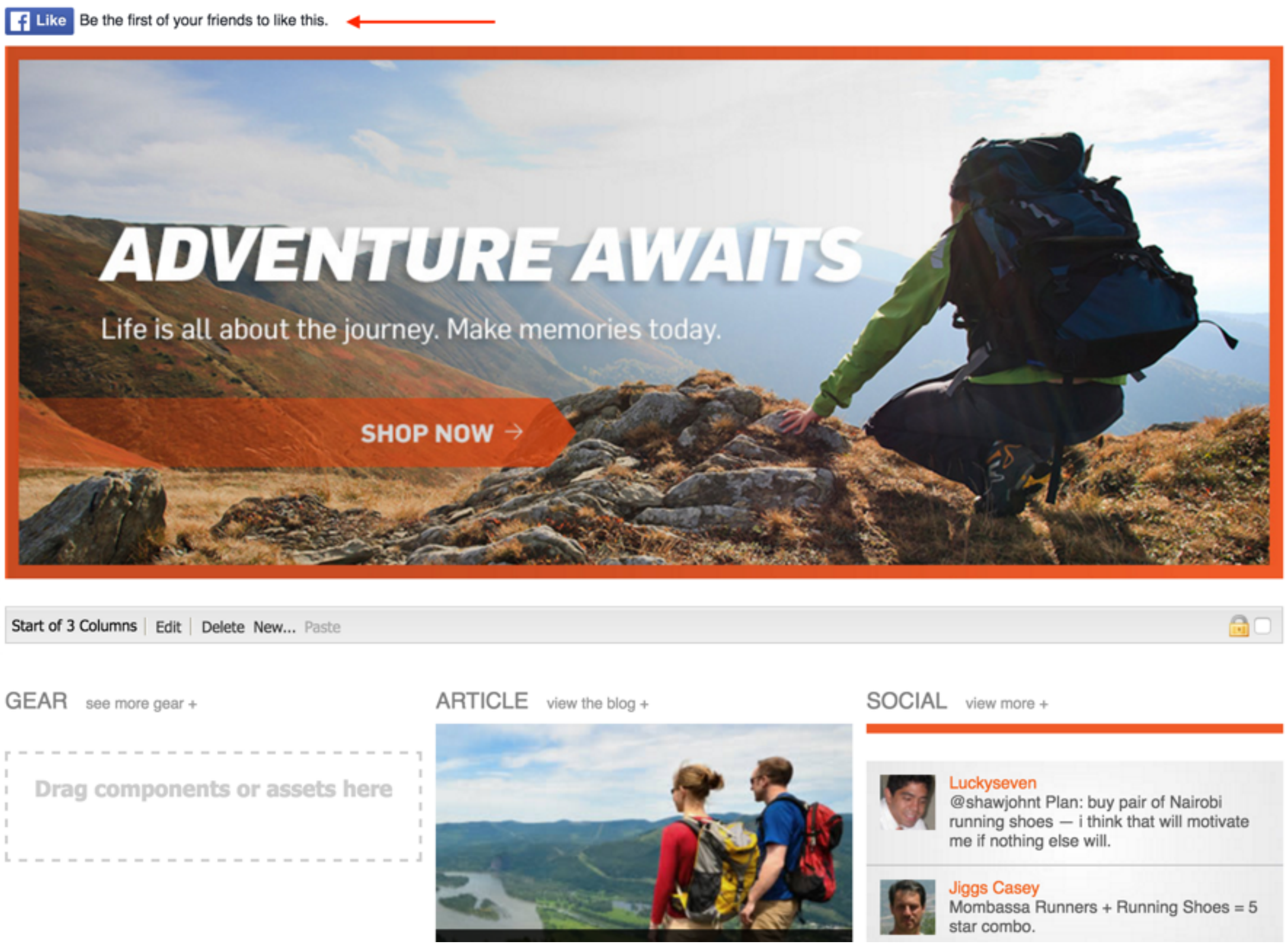
Restart Tealium Configuration Service

1. Go to OSGi "Components" console: http://localhost:4502/system/console/components
2. Lookup for "com.tealium.services.TealiumConfigurationService"
3. For found component in "Actions" row click on "Disable" to disable the component
4. Refresh the page, find previously disabled component and click on "Enable"

Verify Plugin

For Tealium's "Facebook Like Button" tag, the following steps can verify the setup.

1. Open the following link: http://localhost:4502/content/tealium-examples/aem/geometrixx-outdoors/en.html
2. In JavaScript/Web Console, Network tab, verify the page is requesting utag.js
3. Verify that the Facebook like button is displayed on the opened page.

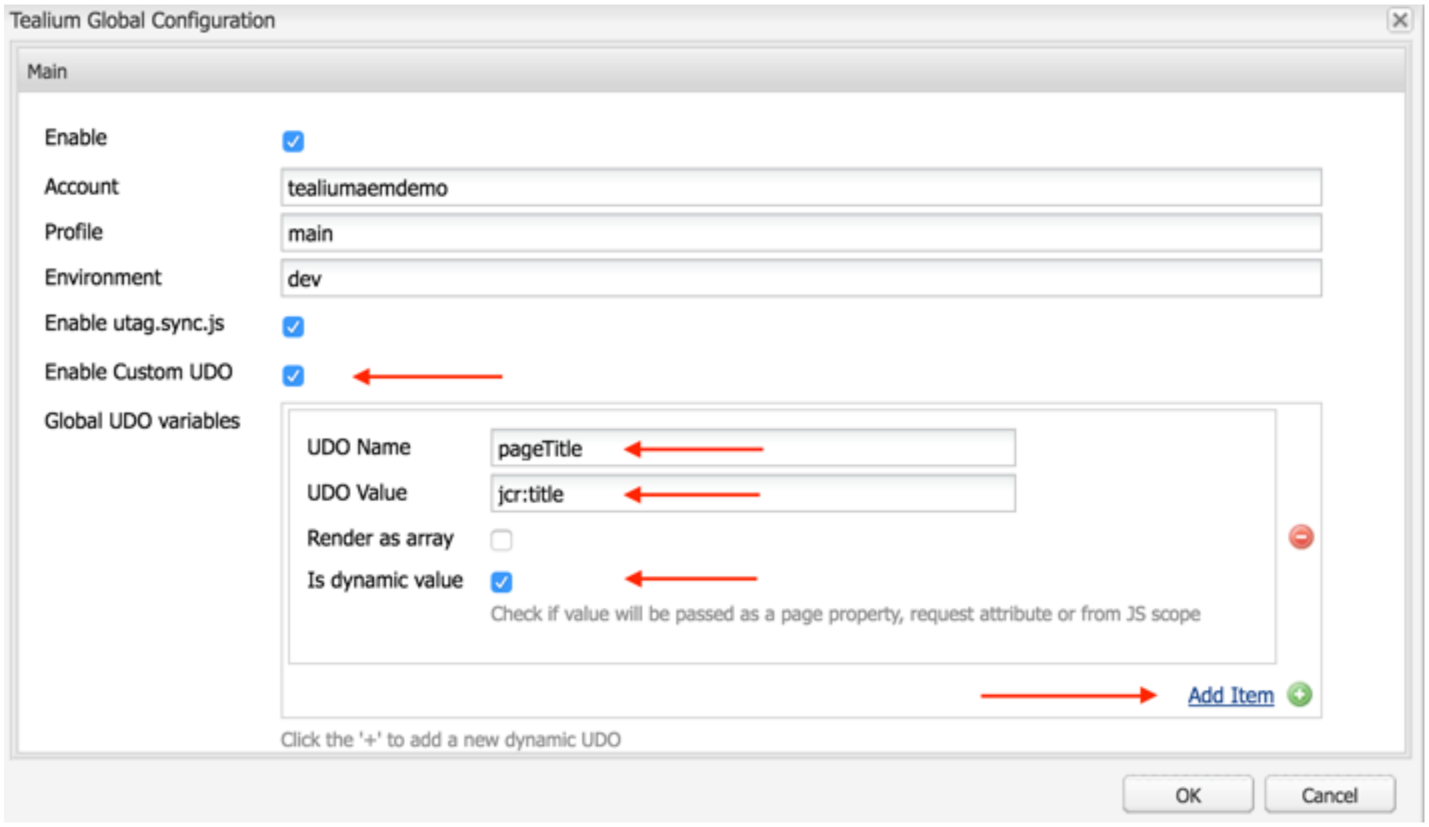

Like
Be the first of your friends to like this.
ADVENTURE AWAITS
Life is all about the journey. Make memories today.
SHOP NOW →
Start of 3 Columns | Edit | Delete New... Paste
GEAR see more gear +
Drag components or assets here
ARTICLE view the blog +
SOCIAL view more +
Luckyseven
@shawjohnt Plan: buy pair of Nairobi running shoes — i think that will motivate me if nothing else will.
Jiggs Casey
Mombassa Runners + Running Shoes = 5 star combo.

Custom Universal Data Object (UDO) Attribute

The following shows how to add custom items to the utag_data object in the page

1. Go to configuration page: http://localhost:4502/cf#/content/tealium-plugin/configuration-page.html
2. Open "Global Configuration Component" configuration dialog
3. Enable checkbox for "Enable Custom UDO"
4. Click on "Add Item"

5. In "UDO Name" enter "pageTitle". If in Tealium iQ you entered different "Source" for load rule (such as "page_title"), please enter that value
6. In "UDO Value" enter ‘jcr:title’
7. Check "Is dynamic value" checkbox. This means that variable's value will be taken from the dynamic scope. In current case this is a name of page property.


Tealium Global Configuration

Main

Enable	✓
Account	tealiumaemdemo
Profile	main
Environment	dev
Enable utag.sync.js	✓
Enable Custom UDO	✓
Global UDO variables	

UDO Name	pageTitle
UDO Value	jcr:title
Render as array	☐
Is dynamic value	✓

Check if value will be passed as a page property, request attribute or from JS scope

Click the '+' to add a new dynamic UDO

8. To check page properties of the example page you can send a GET request to Default Get Servlet with extension ‘.json’ instead of ‘.html’ to get page properties (works only if JSON rendering is enabled): http://localhost:4502/content/tealium-examples/aem/geometrixx-outdoors/en/_jcr_content.json
9. Open http://localhost:4502/content/tealium-examples/aem/geometrixx-outdoors/en.html and verify that "pageTitle" is in the utag_data object with the value of "English"

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. The final set of UDO properties will contain all properties defined in the both global configuration and page type configuration of that specific page template. 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 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 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.