Tealium EPiServer Tag Management Plugin Specification

Version 1.0

Contents

1.0 Overview 3
1.1 Description 3
1.2 Compatibility 3
2.0 Installation 3
2.1 Package Contents 3
2.2 Installation Process 4
3.0 Configuration 4
3.1 Main Settings 4
3.2 Advanced Settings 5
3.3 Layout Injections 6
3.4 Utag Data Settings 6
3.4.1 Common Utag Data 6
3.4.2 Content Type Tags 7
3.4.3 Custom UDO and Computed Fields Mappings 8

1.0 Overview

This document describes how to install, configure, customize and use the **Tealium Tag Management Plugin** for **EPiServer**.

1.1 Description

The plugin represents an object-oriented framework that allows EPiServer developers to embed the **Tealium Tag Management** scripts into their solutions. The plugin is fully configurable and customizable.

1.2 Compatibility

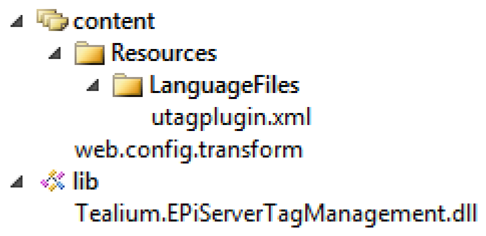
- EPiServer CMS 7 and higher
- .NET Framework 4.0, 4.5

2.0 Installation

The **"Tealium.EPiServerTagManagement.nupkg"** NuGet package is a well-formed EPiServer plugin that can be installed on CMS sites with the Commerce module installed. If the Commerce module is not installed, the **"Tealium.EPiServerTagManagement.Commerce.nupkg"** NuGet package should be installed as well.

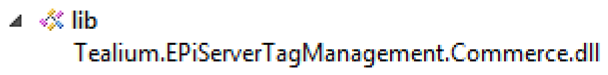
2.1 Package Contents

The **"Tealium.EPiServerTagManagement.nupkg"** package installer adds the following files and items to the website:



```
content
  Resources
    LanguageFiles
      utagplugin.xml
  web.config.transform
lib
  Tealium.EPiServerTagManagement.dll
```

The **"Tealium.EPiServerTagManagement.Commerce.nupkg"** package installer adds one more library:



```
lib
  Tealium.EPiServerTagManagement.Commerce.dll
```

2.2 Installation Process

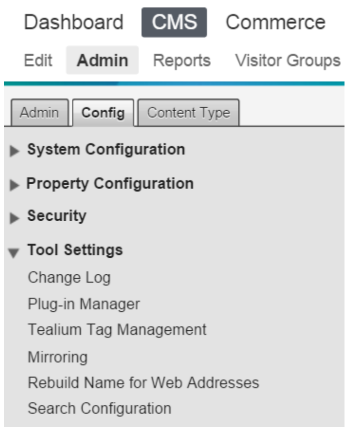
The plugin can be installed from the EPiServer NuGet Feed (http://nuget.episerver.com/) by following the steps below:

> Install-Package Tealium.EPiServerTagManagement

> Install-Package Tealium.EPiServerTagManagement.Commerce

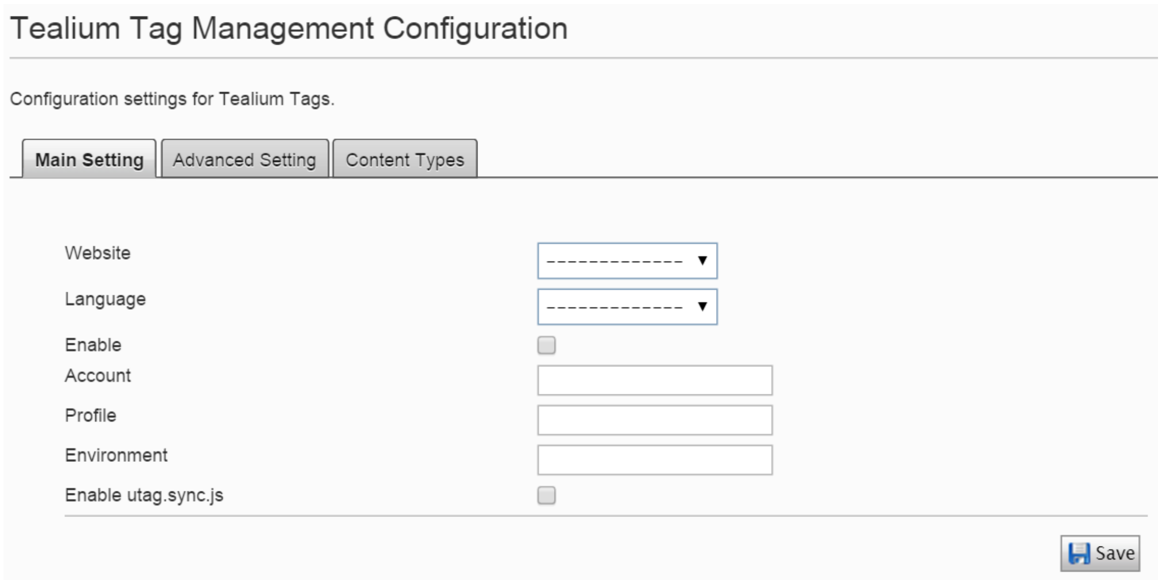
3.0 Configuration

The plugin is available in the **"Tool Settings"** section in the **"Admin"** mode. Please see a screenshot below:



3.1 Main Settings

The main settings can be changed on the **"Tool Settings -> Tealium Tag Management -> Main Settings"** screen. Please see a screenshot below:


Tealium Tag Management Configuration

Configuration settings for Tealium Tags.

Website	------------- ▾
Language	------------- ▾
Enable	☐
Account	
Profile	
Environment	
Enable utag.sync.js	☐

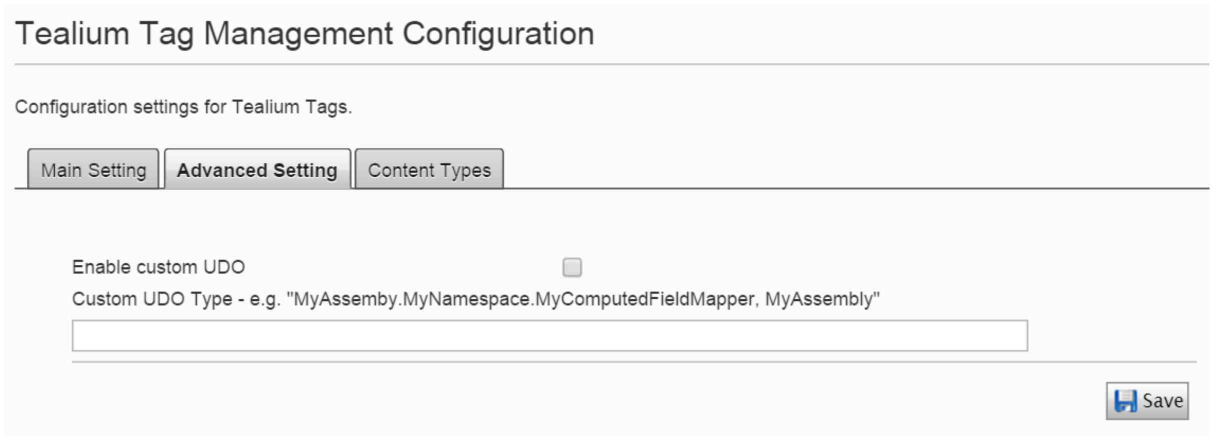
The following settings can be configured:

- **Website:** a list of available websites in the current instance of EPiServer (settings for each website can be configured separately)
- **Language:** a list of available languages in the current instance of EPiServer (settings for each language of a particular website can be configured separately)
- **Enable:** enables or disables the plugin on the whole website
- **Account:** your **"Tealium Account ID"** for connecting with Tealium
- **Profile:** your **"Tealium Profile ID"** for connecting with Tealium
- **Environment:** your **"Tealium Environment ID"** for connecting with Tealium
- **Enable utag.sync.js:** enables or disables the **"utag.sync.js"** script

Please save your changes after you enter all settings.

3.2 Advanced Settings

The advanced settings can be changed on the **"Tool Settings -> Tealium Tag Management -> Advanced Settings"** screen. Please see a screenshot below:



Tealium Tag Management Configuration

Configuration settings for Tealium Tags.

Enable custom UDO

Custom UDO Type - e.g. "MyAssemby.MyNamespace.MyComputedFieldMapper, MyAssembly"

The following settings can be configured:

- **Enable Custom UDO:** enables or disables extending the UDO variable (utag_data) with additional name / value pairs without doing any modifications to the plugin library code
- **Custom UDO Type:** a qualified .NET assembly / type name of a custom computed fields mapping implementation

Please save your changes after you enter all settings.

Please see the "Custom UDO and Computed Fields Mappings" section for more information about dealing with Custom UDO properties.

3.3 Layout Injections

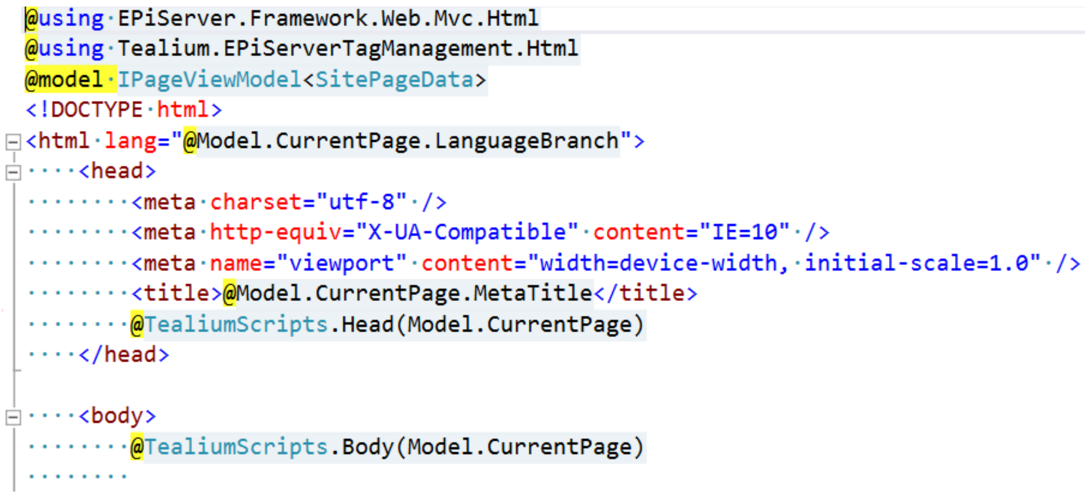
In order to include Tealium scripts and the **"utag_data"** variable into your website pages, please add the following code to your layout files:

- `@TealiumScripts.Head(Model.CurrentPage)`
  This piece of code should be included in the **"head"** section.

- `@TealiumScripts.Body(Model.CurrentPage)`
  This piece of code should be included in the **"body"** section.

*Note: **"Model.CurrentPage"** is your **"IContent"** item.*

Please see an example below:



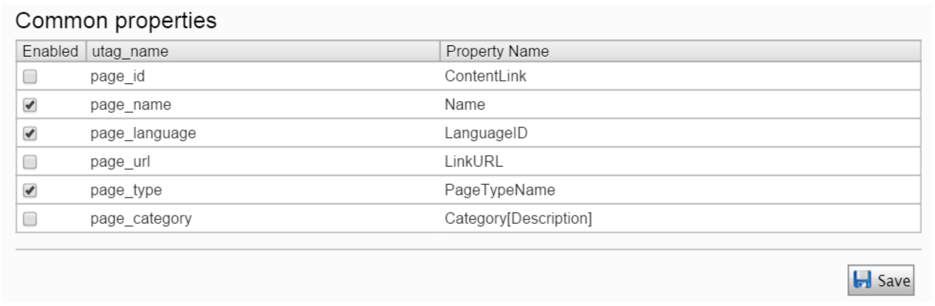
```
@using EPiServer.Framework.Web.Mvc.Html
@using Tealium.EPiServerTagManagement.Html
@model IPageViewModel<SitePageData>
<!DOCTYPE html>
<html lang="@Model.CurrentPage.LanguageBranch">
    <head>
        <meta charset="utf-8" />
        <meta http-equiv="X-UA-Compatible" content="IE=10" />
        <meta name="viewport" content="width=device-width, initial-scale=1.0" />
        <title>@Model.CurrentPage.MetaTitle</title>
        @TealiumScripts.Head(Model.CurrentPage)
    </head>

    <body>
        @TealiumScripts.Body(Model.CurrentPage)
        
```

3.4 Utag Data Settings

3.4.1 Common Utag Data

In order to configure parameters that should be rendered on all website pages, please open **"Tool Settings -> Tealium Tag Management -> Main Settings"**. Please see a screenshot below:



Common properties

Enabled	utag_name	Property Name
☐	page_id	ContentLink
✓	page_name	Name
✓	page_language	LanguageID
☐	page_url	LinkURL
✓	page_type	PageTypeName
☐	page_category	Category[Description]

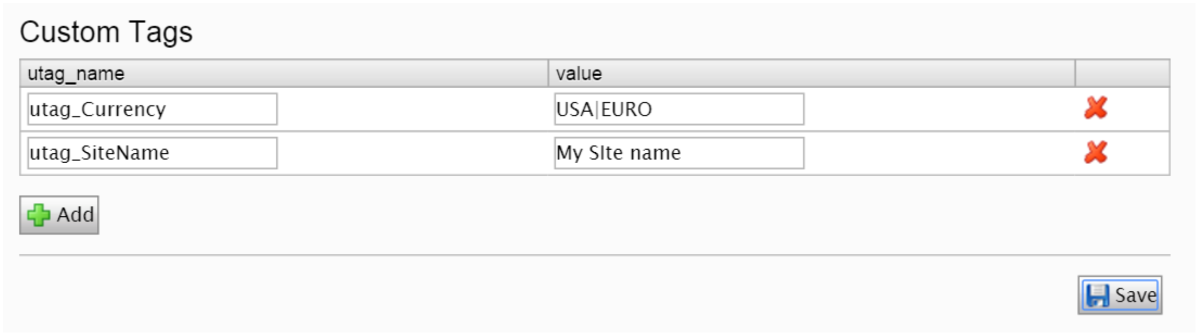
Please check all necessary check boxes and save your changes.

The final JS code might look like:

```
        <script type='text/javascript'
src='//tags.tiqcdn.com/utag/test_account2/test_profile2/test_environment2/utag.sync.js'></script>
    </head>

    <body>
        <script type="text/javascript"> var utag_data = {
page_id: "164",
page_name: "StartDK",
page_language: "da",
page_url: "http://samplealloysitedk/",
};
</script>
<script type="text/javascript"> (function(a,b,c,d){
a='//tags.tiqcdn.com/utag/test_account2/test_profile2/test_environment2/utag.js'; b=document; c='script';
d=b.createElement(c); d.src=a; d.type='text/java'+c; d.async=true; a=b.getElementsByTagName(c)[0];
a.parentNode.insertBefore(d,a); })(); </script>
```

The **“Custom Tags”** section allows you configuring custom name / value pairs that should be added to the **“utag_data”** variable on all website pages. Please see an example below:


Custom Tags

utag_name	value	
utag_Currency	USA|EURO	
utag_SiteName	My SIte name	

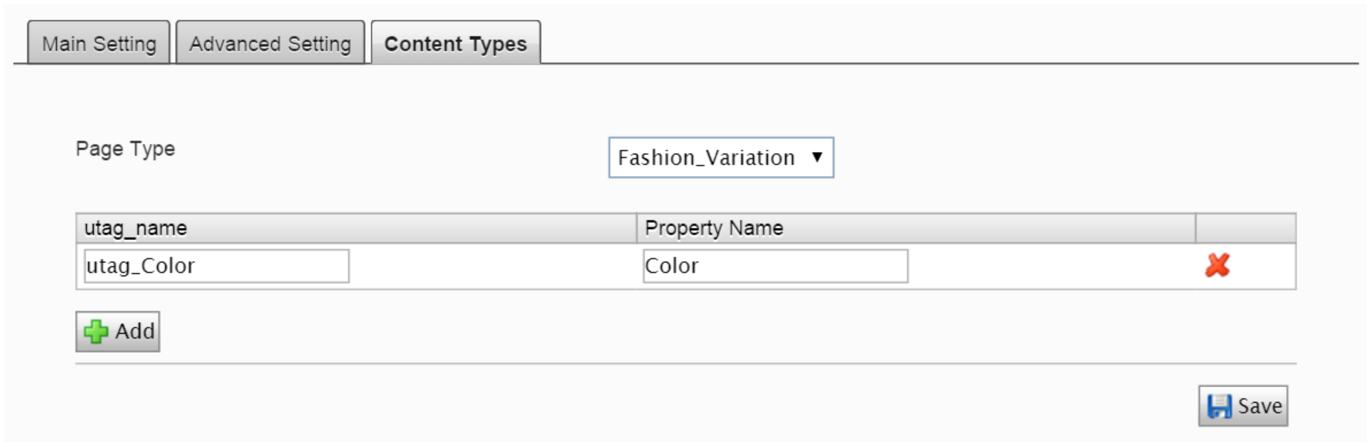
```
        <script type="text/javascript"> var utag_data = {
page_id: "164",
page_name: "StartDK",
page_language: "da",
page_url: "http://samplealloysitedk/",
utag_currency: ["USA","EURO"],
utag_sitename: "My site name"
};
```

Note: In order to enter multiple values for a custom property, please separate those values with “|”; in this case, it will be rendered as a JS array.

Please save your changes after you finish configuring custom properties.

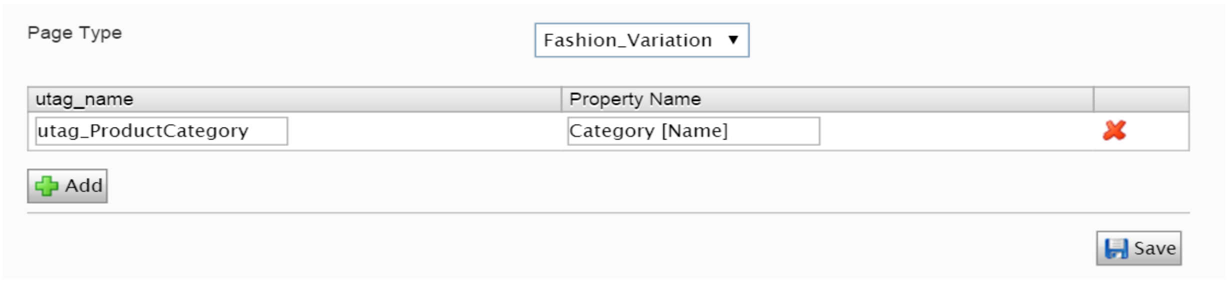
3.4.2 Content Type Tags

The plugin allows configuring custom properties separately per each EPiServer page type or commerce meta class. The **“Page Type”** drop-down box on the “**Tool Settings -> Tealium Tag Management -> Content Types**” screen contains a list of available content types in the system (including the commerce ones). Please see a screenshot below:



Page Type	Fashion_Variation ▼	
utag_name	Property Name	
utag_Color	Color	✗

*Note: In order to render a value of a particular property of a page type object, please use the following simple syntax: **"Object[Property]"**. For example, **"Category[Name]"**.*



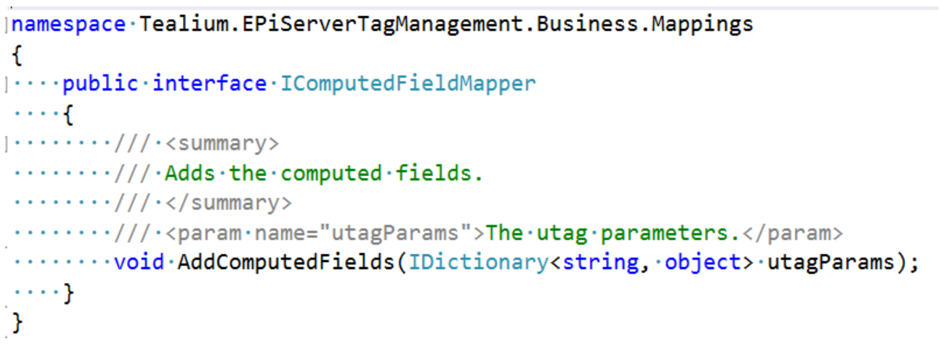
Page Type	Fashion_Variation ▼	
utag_name	Property Name	
utag_ProductCategory	Category [Name]	✗

Please save your changes once you are done with configuring properties for a specific page type.

3.4.3 Custom UDO and Computed Fields Mappings

The plugin allows EPiServer developers to implement a custom property mapping to render any kind of properties on a specific page or on all website pages.

First, you need to implement the **"IComputedFieldMappings"** interface, which is declared as follows:



```
namespace Tealium.EPiServerTagManagement.Business.Mappings
{
    public interface IComputedFieldMapper
    {
        /// <summary>
        /// Adds the computed fields.
        /// </summary>
        /// <param name="utagParams">The utag parameters.</param>
        void AddComputedFields(IDictionary<string, object> utagParams);
    }
}
```

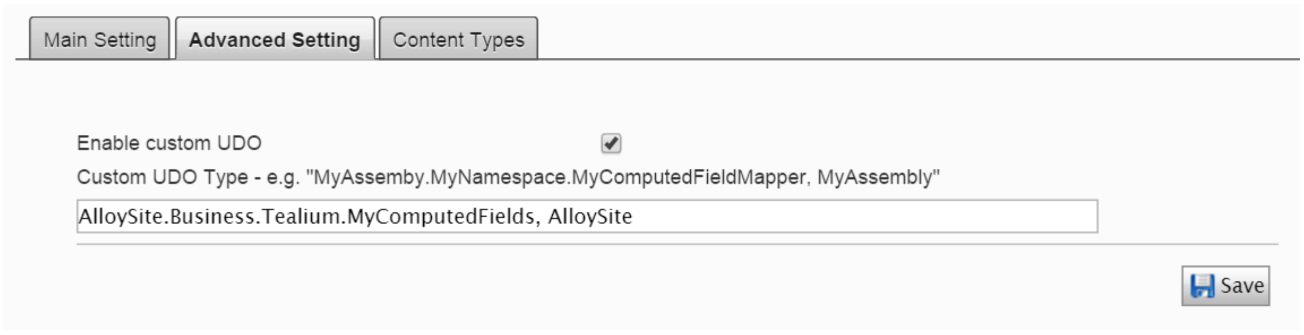
Please see below an example of how to render a search term on all website pages:

```
namespace AlloySite.Business.Tealium
{
    public class MyComputedFields : IComputedFieldMapper
    {
        public void AddComputedFields(IDictionary<string, object> utagParams)
        {
            this.AddSearchTerm(utagParams);
        }

        private void AddSearchTerm(IDictionary<string, object> utagParams)
        {
            var searchTerm = HttpContext.Current.Request["query"];

            if (searchTerm != null)
            {
                if (!utagParams.ContainsKey("search_term"))
                {
                    utagParams.Add("search_term", searchTerm);
                }
            }
        }
    }
}
```

Second, you need to go to **"Tool Settings -> Tealium Tag Management -> Advanced Settings"**, check the **"Enable Custom UDO"** check box and specify a qualified type name of your custom fields mapping implementation. Please see a screenshot below:



Enable custom UDO ✓
Custom UDO Type - e.g. "MyAssemby.MyNamespace.MyComputedFieldMapper, MyAssembly"
AlloySite.Business.Tealium.MyComputedFields, AlloySite

The result JS code might look like (if a query string parameter is **"?query=test"**):

```
        <script type="text/javascript"> var utag_data = {
page_id: "164",
page_name: "StartDK",
page_language: "da",
page_url: "http://samplealloysitedk/",
utag_currency: ["USA","EURO"],
utag_sitename: "My site name",
search_term: "test"
};
</script>
```

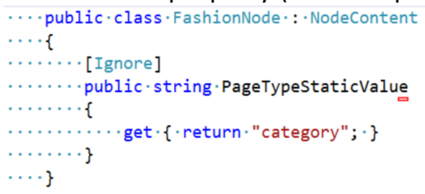
The plugin uses the following order while resolving values of each parameters to be rendered:

1. Common Parameters.
2. Custom Parameters.
3. Page Type Parameters.
4. Computed Fields.

Note: Custom Parameters override Common Parameters (if any conflicts), Page Type Parameters override Custom and Common Parameters, Computed Fields override all other parameters in case of any conflict.

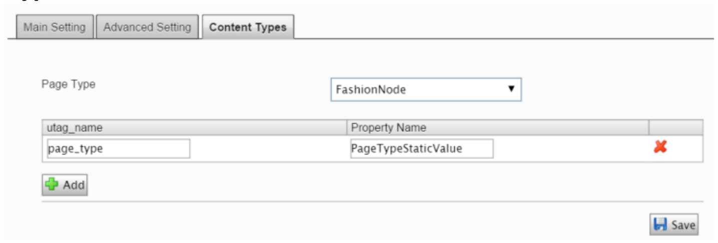
For example, if you would like to override the **"page_type"** common parameter with a custom value for all instances of the **"Commerce Node"** page type, you might need to:

1. Create a new property (for example, **"PageTypeStaticValue"**):



```
    public class FashionNode : NodeContent
    {
        [Ignore]
        public string PageTypeStaticValue
        {
            get { return "category"; }
        }
    }
```

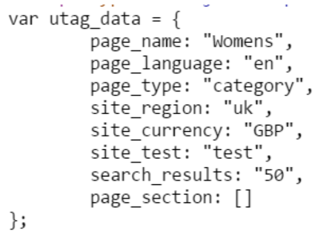
2. Configure a proper name / value property on the "**Tool Settings -> Tealium Tag Management -> Content Types**" screen.



Page Type: FashionNode

utag_name	Property Name
page_type	PageTypeStaticValue

3. That's it. The result JS code might look like:



```
var utag_data = {
        page_name: "Womens",
        page_language: "en",
        page_type: "category",
        site_region: "uk",
        site_currency: "GBP",
        site_test: "test",
        search_results: "50",
        page_section: []
};
```