EPiServer Google Analytics for developer guideline

[Introduction](#h.f2pwiiz5c4yp)

[Inject custom tracking script into a page](#h.t10in4y4zor5)

[Decide where to put tracking script](#h.lh5q8so2ah9k)

[Tracking custom data with GA](#h.nb949jt419hm)

[Override to build tracking script](#h.5pwj7v9jg877)

[Tracking Commerce with GA](#h.xdqcjsi601rg)

[Example A : HOW TO integrate e-commerce tracking on EPiServer Commerce Sample site](#h.madi2iv3f5jy)

[EXAMPLE B: HOW TO integrate tracking on a site that redirects to a thank you page after purchase](#h.lmipsa6gcymr)

[Using Enhanced Commerce mode](#h.66l26wjzx7cj)

# Introduction

Google Analytics Add-On (GA) now has some refactoring and improvement to be more developer friendly and easier for customization.

* All of API call to Google Analytics API from our GA is now made by RESTful API, we are not bound to Google's DLLs, so it is easier to deploy.
* Part of GA is refactor, with Dependency Injection in mind
* New features bring a lot of useful contextual information for a page
* A few interfaces and flexible points give more room for 3rd party customized code

So here we write a technical documentation and provide more detail examples of how to use GA with your website in developer oriented way, especially integrate with Commerce site, using both **normal commerce** and **enhanced commerce** mode of Google Analytics service

# Inject custom tracking script into a page

There is an interface in EPiServer GA core, IPluginScript , which we can use to inject additional tracking script to public ViewMode. One example is the EPiServer.GoogleAnalytics.Commerce uses it to load e-commerce plugin :

public string GetScript()

{

if (\_trackerSettings().TrackingScriptOption == TrackingScriptOption.Universal)

{

return "ga('require', 'ecommerce');";

}

return string.Empty;

}

Install **EPiServer.GoogleAnalytics.Commerce** nuget package(to deploy EPiServer.GoogleAnalytics.Commerce.dll to folder “bin” of the website) and make sure you choose Google Universal Analyticsadmin page, the result is the required script is injected to tracking code at page's header.



# Decide where to put tracking script

In case you decide specified pages that will be running tracking script, or you want to modify script before sending the data. For instance, you want to delay running command ga('send', 'pageview'), you can get the tracking script by calling below extension methods (in namespace EPiServer.GoogleAnalytics.Helpers):

For web form: public static string GetAnalyticsTrackingScript(this PageBase page, HttpContext context)

For MVC: public static MvcHtmlString AnalyticsTrackingScript(this System.Web.Mvc.HtmlHelper html, PageData currentPage)

Those method returns script with the configured syntax (Classic or UA) that you set in **GA Settings** page. You can build the script by specify syntax you would like to have as sample code below:

var scriptBuilder = new ScriptBuilder(EPiServer.Web.SiteDefinition.Current, () =>

{

return new SiteTrackerSettings

{

TrackingScriptOption = TrackingScriptOption.Universal, // specify syntax you want to generate tracking script

TrackingId = "UA-56173669-1"

};

});

var requiresScriptReference = false;

var context = new HttpContextWrapper(HttpContext.Current);

var script = scriptBuilder.BuildTrackingScript(out requiresScriptReference, context, CurrentPage);

// now you have tracking script with UA syntax, you can modify or put it into your pages

NOTE: You should go to AdminMode. change to use Custom Script and let it empty. If not, duplicated script section will be rendered.

# Tracking custom data with GA

1. Install **EPiServer.GoogleAnalytics** nuget package (Add reference to EPiServer.GoolgeAnalytics.dll)
2. Add the following as C# code to track custom data

// Add using directive on the top of cs file

using EPiServer.GoogleAnalytics.Helpers;

// Register custom tracking

var paramsObject = new Dictionary<string, object>();

paramsObject.Add("dimention5", "custom dimension data");

paramsObject.Add("metric5", "custom metric data");

Context.AddAnalyticsCustom("set", true, paramsObject);

Since those tracking code was added, the tracking script will be generate somethings like below and the custom data will be sent to Google Analytic service.:

ga('set', {

'dimension5': 'custom dimension data',

'metric5': 'custom metric data'

});

# Override to build tracking script

If you need to override of building a tracking script type, example: you are using Universal Analytics and want to override BuildTrackingScript method to inject your own script into.

1. Create a class inherit from GASyntax class you would like to override:

public class MyCustomUASynctax : UniversalSyntax

{

public override string BuildTrackingScript(ScriptBuilderContext appenderContext, GoogleAnalytics.Models.SiteTrackerSettings siteSettings, out bool requiresScriptReference)

{

var script = base.BuildTrackingScript(appenderContext, siteSettings, out requiresScriptReference);

// add/modify tracking script here

return script;

}

}

1. Register your class with ServiceLocator to use your class instead of UniversalSyntax

Create InitializableModule which implement IConfigurableModule and depend on FrameworkInitialization like below:

[InitializableModule]

[ModuleDependency(typeof(FrameworkInitialization))]

public class InitializationModule : IConfigurableModule

{

/// <summary>

/// Setup StructureMap configuration, auto called in website start

/// </summary>

/// <param name="context"></param>

public void ConfigureContainer(ServiceLocation.ServiceConfigurationContext context)

{

context.Container.Configure(c => c.For<UniversalSyntax>().Use<MyCustomUASynctax>());

}

}

From now on, the add-on will use MyCustomUASynctax for building script with using UA option, and you can add/modify the script in your BuildTrackingScript method.

# Tracking Commerce with GA

A checkout process typically ends with a **Thank You** page. This is a good place to track purchases with GA. Let’s say that you are using Seagull template, the final step can retrieve purchase information via an AJAX call and do the tracking on the client (EXAMPLE A). If the checkout ends with a redirect to a **Thank You** page, this page can retrieve the **order number** and track it with GA via an extension method on Context (EXAMPLE B).

Don't forget to enable e-commerce tracking at [http://google.com/analytics/ below admin/profiles/profile settings/](http://google.com/analytics/)

**These examples are demonstrated with EPiServer Commerce Sample template**

On Commerce Sample template site, after the Checkout was processed there are 2 circumstances:

* The customer has been login then will be redirected to the **Order** administration page
* The customer is anonymous then will be redirected to the **CheckoutConfirmationStep** page

This below guide is for the anonymous user case, will be redirected to the **CheckoutConfirmationStep** page after the order was processed (for logged in user the examples code below should be added in **Order.aspx** and **Order.aspx.cs**)

## Example A : HOW TO integrate e-commerce tracking on EPiServer Commerce Sample site

1. Install **EPiServer.GoogleAnalytics.Commerce** nuget package (Add reference to EPiServer.GoolgeAnalytics.Commerce.dll)

2. Edit /Templates/Sample/Pages/CheckoutConfirmationStep.aspx

2.1. Add import namespace directive on top of the file

<%@ Import Namespace="System.Linq" %>

<%@ Import Namespace="EPiServer.GoogleAnalytics.Helpers" %>

<%@ Import Namespace="Mediachase.Commerce.Security" %>

2.2. Add tracking code to the page template:

1. If you are tracking with **Google Classic Analytics**, add following code

<%

var order = Mediachase.Commerce.Orders.OrderContext.Current.GetPurchaseOrders(PrincipalInfo.CurrentPrincipal.GetContactId()).OrderBy(o => o.OrderGroupId).LastOrDefault();

var trackingNumber = order != null ? order.TrackingNumber.ToString(System.Globalization.CultureInfo.InvariantCulture) : string.Empty;

%>

<script type="text/javascript">

$.getJSON("/CommerceTracking/GetOrderTransaction/<%=trackingNumber %>", function (data) {

// The result contains an isSuccess which is true when a purchase order was found

if (!data.isSuccess || typeof \_gaq === "undefined") {

return;

}

// The result contains a list of tracking arrays that can be pushed directly to the google tracking API

data.transaction.forEach(function (item) {

\_gaq.push(item);

});

// Sends the tracking information to google

\_gaq.push(['\_trackTrans']);

});

</script>

1. If you are tracking with **Google Universal Analytics**, add following code

<%

var order = Mediachase.Commerce.Orders.OrderContext.Current.GetPurchaseOrders(PrincipalInfo.CurrentPrincipal.GetContactId()).OrderBy(o => o.OrderGroupId).LastOrDefault();

var trackingNumber = order != null ? order.TrackingNumber.ToString(System.Globalization.CultureInfo.InvariantCulture) : string.Empty;

%>

<script type="text/javascript">

ga('require', 'ecommerce');

$.getJSON("/CommerceTracking/GetOrderTransaction/<%=trackingNumber %>", function (data) {

// The result contains an isSuccess which is true when a purchase order was found

if (!data.isSuccess || typeof ga === "undefined") {

return;

}

data.transaction.forEach(function (item) {

if (item[0] == "\_addTrans") {

ga('ecommerce:addTransaction', {

'id': item[1], // Transaction ID. Required.

'affiliation': item[2], // Affiliation or store name.

'revenue': item[3], // Grand Total.

'shipping': item[5], // Shipping.

'tax': item[4], // Tax.

'currency': item[9] // Currency.

});

} else if (item[0] == "\_addItem") {

ga('ecommerce:addItem', {

'id': item[1], // Transaction ID. Required.

'name': item[3], // Product name. Required.

'sku': item[2], // SKU/code.

'category': item[4], // Category or variation.

'price': item[5], // Unit price.

'quantity': item[6] , // Quantity.

'currency': item[7] // Currency.

});

}

});

// Sends the tracking information to google

ga('ecommerce:send');

});

</script>

## EXAMPLE B: HOW TO integrate tracking on a site that redirects to a thank you page after purchase

1. Install **EPiServer.GoogleAnalytics.Commerce** nuget package (Add reference to EPiServer.GoolgeAnalytics.Commerce.dll)
2. Add the following to page load of file CheckoutConfirmationStep.aspx.cs:  
   Add using directive on the top of file:  
     
   using EPiServer.GoogleAnalytics.Commerce.Helpers;  
   // Registers the order to be tracked by the google analytics tracking script added by the module, and generate tracking script on current page only  
   Context.AddAnalyticsTransaction(order, CurrentPage.ContentLink);
3. Rebuild the project

**Example C: HOW TO integrate e-commerce tracking on QuickSilver site**

* + - 1. Install **EPiServer.GoogleAnalytics.Commerce** nuget package (Add reference to EPiServer.GoolgeAnalytics.Commerce.dll)
      2. Use **Google Universal Analytics** in GA settings
      3. Open **OrderConfirmationController.cs** then insert following code

using EPiServer.ServiceLocation;

using EPiServer.Framework.Web.Resources;

…

and before **return** statement of **Index** method:

var trackingScript = @"ga('require', 'ecommerce');

$.getJSON('/CommerceTracking/GetOrderTransaction/" + order.TrackingNumber + @"', function (data) {

// The result contains an isSuccess which is true when a purchase order was found

if (!data.isSuccess || typeof ga === 'undefined') {

return;

}

data.transaction.forEach(function (item) {

if (item[0] == '\_addTrans') {

ga('ecommerce:addTransaction', {

'id': item[1], // Transaction ID. Required.

'affiliation': item[2], // Affiliation or store name.

'revenue': item[3], // Grand Total.

'shipping': item[5], // Shipping.

'tax': item[4], // Tax.

'currency': item[9] // Currency.

});

} else if (item[0] == '\_addItem') {

ga('ecommerce:addItem', {

'id': item[1], // Transaction ID. Required.

'name': item[3], // Product name. Required.

'sku': item[2], // SKU/code.

'category': item[4], // Category or variation.

'price': item[5], // Unit price.

'quantity': item[6] , // Quantity.

'currency': item[7] // Currency.

});

}

});

// Sends the tracking information to google

ga('ecommerce:send');

});";

var requiredResources = ServiceLocator.Current.GetInstance<IRequiredClientResourceList>();

requiredResources.RequireScriptInline(trackingScript, "GACommerce", new string[] { }).AtFooter();

**Example D: HOW TO integrate e-commerce tracking on QuickSilver site using GA helper methods**

* + - 1. Install **EPiServer.GoogleAnalytics.Commerce** nuget package (Add reference to EPiServer.GoolgeAnalytics.Commerce.dll)
      2. Open **OrderConfirmationController.cs** then insert following code

using EPiServer.GoogleAnalytics.Commerce.Helpers;

and before return View(viewModel); statement of **Index** method:

// Registers the order to be tracked by the google analytics tracking script added by the module, and generate tracking script on current page only

System.Web.HttpContext.Current.AddAnalyticsTransaction(order as PurchaseOrder, currentPage.ContentLink);

# Using Enhanced Commerce mode

This is an example to use the Enhanced Commerce (ec.js) library. This supposes you implement your code in your website project (normally you get and directly make changes to the **EPiServer.Commerce.Sample** code)

1. Inherit from CommercePluginScript class and override the GetScript method to **require** the **enhanced commerce** and add some more code:

[ServiceConfiguration(typeof(IPluginScript))]

public class EnhancedCommercePlugin : CommercePluginScript //IPluginScript

{

private Factory<SiteTrackerSettings> \_trackerSettings;

public EnhancedCommercePlugin(Factory<SiteTrackerSettings> trackerSettings)

: base(trackerSettings)

{

\_trackerSettings = trackerSettings;

}

public override string GetScript()

{

if (\_trackerSettings().TrackingScriptOption != TrackingScriptOption.Universal)

{

return string.Empty;

}

return "ga('anoynymoizeIP', true); ga('require', 'ec');";

}

}

1. Override to build your own custom syntax:

public class MyCustomUASynctax : UniversalSyntax

{

public override string BuildTrackingScript(ScriptBuilderContext appenderContext, GoogleAnalytics.Models.SiteTrackerSettings siteSettings, out bool requiresScriptReference)

{

var script = base.BuildTrackingScript(appenderContext, siteSettings, out requiresScriptReference);

// add/modify tracking script here

return script;

}

}

1. Register your classes with ServiceLocator:

[InitializableModule]

[ModuleDependency(typeof(FrameworkInitialization))]

public class InitializationModule : IConfigurableModule

{

...

/// <summary>

/// Setup StructureMap configuration, auto called in website start

/// </summary>

/// <param name="context"></param>

public void ConfigureContainer(ServiceLocation.ServiceConfigurationContext context)

{

context.Container.Configure(c =>

{

c.For<UniversalSyntax>().Use<MyCustomUASynctax>();

c.For<IPluginScript>().Use<EnhancedCommercePlugin>();

});

}

...

}

1. Implement the methods following the **Enhanced Commerce** syntax:

public class Trackings

{

public void TrackProductImpression(HttpContextBase context, string code, string name, string category = null,

string brand = null, string variant = null, string list = null, int position = 0)

{

Dictionary<string, object> dict = new Dictionary<string, object>();

if (code != null) dict.Add("id", code);

if (name != null) dict.Add("name", name);

if (category != null) dict.Add("category", category);

if (brand != null) dict.Add("brand", brand);

if (variant != null) dict.Add("variant", variant);

if (list != null) dict.Add("list", list);

if (position > 0)

dict.Add("position", position.ToString());

context.AddAnalyticsCustom("ec:addImpression", true, dict);

}

public void TrackProductDetail(HttpContextBase context, string code, string name, string path, string category = null, string brand = null, int price = 0)

{

var dict = new Dictionary<string, object>();

if (code != null) dict.Add("id", code);

if (name != null) dict.Add("name", name);

if (category != null) dict.Add("category", category);

if (brand != null) dict.Add("brand", brand);

if (price > 0) dict.Add("price", price.ToString());

context.AddAnalyticsCustom("ec:addProduct", true, dict);

context.AddAnalyticsCustom("ec:setAction", true, "detail");

}

public void TrackTransaction(HttpContextBase context, PurchaseOrder order)

{

var orderForm = order.OrderForms[0];

foreach (LineItem item in orderForm.LineItems)

{

context.AddAnalyticsInteraction(GetAddProduct(item));

}

var orderId = order.TrackingNumber;

var dict = new Dictionary<string, object>();

dict.Add("id", order.TrackingNumber);

dict.Add("affiliation", CommerceExtensions.GetAffiliation(order.AffiliateId));

dict.Add("revenue", order.Total);

dict.Add("tax", order.TaxTotal);

dict.Add("shipping", order.ShippingTotal);

context.AddAnalyticsCustom("ec:setAction', 'purchase", true, dict);

}

private AnalyticsInteraction GetAddProduct(LineItem item)

{

var dict = new Dictionary<string, object>();

dict.Add("id", item.CatalogEntryId);

dict.Add("name", item.DisplayName);

dict.Add("category", item.CatalogNode);

dict.Add("price", item.PlacedPrice);

dict.Add("quantity", item.Quantity);

return new AnalyticsInteraction("ec:addProduct", dict) { ClearWhenContextChanged = true };

}

}

1. Using the new tracking methods in the **EPiServer.Commerce.Sample** template:

Call TrackTransaction in the CheckoutConfirmationStep.aspx.cs

protected void Page\_Load(object sender, EventArgs e)

{

var order = OrderContext.Current.GetPurchaseOrders(PrincipalInfo.CurrentPrincipal.GetContactId()).OrderBy(o => o.OrderGroupId).LastOrDefault();

if (order == null)

return;

liPONumber.Text = order.TrackingNumber.ToString(CultureInfo.InvariantCulture);

liTotal.Text = new Money(order.Total, order.BillingCurrency).ToString();

var trackings = new Trackings();

trackings.TrackTransaction(new HttpContextWrapper(Context), order);

}