Sitecore CMS 6.5

Sitecore Engagement Analytics API Cookbook

Illustrative Examples for Developers

Table of Contents

Chapter 1

Introduction

This document provides an overview of the Sitecore Engagement Analytics. It covers an overview of the features, the architecture and how to perform common tasks using the Engagement Analytics API.

- **Chapter 1 — Introduction**
  This chapter introduces the main features of Sitecore Engagement Analytics and common terminology.
- **Chapter 2 — Architecture**
  This chapter contains a conceptual overview of the Engagement Analytics architecture.
- **Chapter 3 — Common Tasks in the API**
  This chapter describes some common programming tasks that you can perform with the DMS/Engagement Analytics API.

This chapter contains the following sections:

- Features
- Common Terminology

1.1 Features

Some of the main features of Sitecore Engagement Analytics include:

- Marketing Automation
- Real-Time Personalization
- Dashboards and Reports
- Classification of Visitors
- Campaign Tracking
- Visitor Attributes and Profiles
- Multivariate Testing
- Geo IP Lookup data

This section provides a brief definition and overview of these topics.

More detailed information about these topics can be found in related documents such as the:

- *Marketing Operations Cookbook*
- *Executive Dashboard Cookbook*
- *Engagement Analytics Configuration Reference*
- *Engagement Analytics API Reference*
- *Rules Engine Cookbook*
- *Report Designer Cookbook*

1.1.1 Marketing Automation

Marketing Automation functions much like a workflow for web site visitors. For example, an automation plan might check if a visitor is responding to a specific email campaign and then personalize the content shown based on the specific campaign. If an email recipient does not respond to the original email and visit the site, the automation plan might send a follow-up email. For more information about marketing automation plans, see the *Marketing Operations Cookbook*.

1.1.2 Real-Time Personalization

Real-Time Personalization enables your web site to customize content for each visitor based on the information known about that visitor. This information can include Geo IP data, campaign data, visitor profile data and more.

1.1.3 Dashboards and Reports

The Sitecore Desktop User Interfaces include advanced reporting and status information that allow business users to easily see common reports about web site traffic. Please refer to the *Executive Dashboard Cookbook* for more information.

1.1.4 Classification of Visitors

Sitecore Engagement Analytics provides automatic visitor classification based on information that is known about each visit such as Geo IP, search keywords and search type, referrers and profile attributes.

1.1.5 Campaign Tracking

Sitecore Engagement Analytics enables tracking of web site visits that are responding to various advertising campaigns. Campaigns can be tracked by specific URL strings that are published in email and online advertisements. Campaigns can also be associated with landing pages.

1.1.6 Visitor Attributes and Profiles

Visitor attributes and profile information consist of all of the data that is known about a visitor and is accumulated over multiple visits. This information consists of Geo IP data, pages visited, goals and other events that occurred during visits, campaign associations, referrers, and search keywords. Automatic classification of visitors is possible by matching up various attributes of a visitor with pre-defined profiles.

Engagement Analytics allows you to assign values to these different attributes. Analytics reports take into account the overall value and effectiveness of each visit and do not just track number of page views.

1.1.7 Multivariate Testing

Multivariate testing enables you to test different versions of your web site content to see which version is most effective in achieving goal conversions and high value, effective visits. For more information about configuring multivariate tests, see the *Marketing Operations Cookbook*.

1.1.8 Geo IP Lookup Data

Geo IP Lookup Data is provided to Sitecore Engagement Analytics through third party web services. Geo IP data includes country, metro area, city, IP address, domain owner, ISP and other related information. Each Sitecore installation includes a trial period for the Geo IP service provider. After the trial period Sitecore customers must contract directly with the Geo IP service provider.

Geo IP lookups are done asynchronously. If there is a delay in connecting with the web service then Geo IP information may not be available immediately for a new visitor. Geo IP data is stored in the Analytics database, so lookups do not have to be performed for returning visitors.

1.2 Common Terminology

Sitecore Engagement Analytics uses the following terms to describe common actions and operations:

1.2.1 Visitor

A person that makes one or more visits to a website.

1.2.2 Visit

All the actions a visitor takes from the moment they enter a website until the moment they leave and consists of all the pages viewed, and resources consumed, such as campaigns triggered and conversions achieved.

1.2.3 Page Event

An action, associated with a webpage, which is triggered either by a visitor or automatically.

Examples:

- Visitor triggered page event – the visitor submits a form.
- Automatically triggered page event – the time limit on the form expires.

Chapter 2

Architecture

This chapter provides a conceptual overview of the Engagement Analytics architecture and database operations.

- Database Operations

2.1 Database Operations

Database operations in Sitecore Engagement Analytics have been optimized in two important ways.

- The API has been expanded to provide convenient methods to access visitor information without having to address the underlying database tables.
- Engagement Analytics data is stored in memory before being written to persistent database storage.

To learn more about the Sitecore Engagement Analytics APIs please review the rest of this document and the *Sitecore Engagement Analytics API Reference*.

When using the Sitecore Engagement Analytics APIs Visitor data may be accessed from memory or from persistent database storage. When a Visitor session begins information about that Visit is first stored in session data storage in server memory. When individual visit data reaches a defined threshold amount, or when a Visit session ends, the data is moved to a shared in memory DataSet. When the shared in memory DataSets reach a defined threshold, based on the amount or data or time elapsed, all of the data is written to persistent storage in the Sitecore Analytics database.

The batch size and timing for persistent database updates are configurable in `web.config`. For more information about optimizing database operations, see the *Engagement Analytics Configuration Reference*.

Chapter 3

Common Tasks in the API

This chapter will provide an introduction to programming common tasks with the DMS/Engagement Analytics API.

- Accessing Visitor Data
- Accessing Tracking Field Data

3.1 Accessing Visitor Data

In order to access Visitor data in the Engagement Analytics API, you primarily use the following classes:

- Sitecore.Analytics.Tracker
- Sitecore.Analytics.TrackerDataContext

Most of the following code samples require these namespace `using` directives:

```
using Sitecore.Analytics;
using Sitecore.Analytics.Data.DataAccess;
using Sitecore.Analytics.Data.DataAccess.DataSets;
```

3.1.1 How to Access the Current Session

The following sample code shows how to access information about the current session. This example shows how to access information about any search engine keywords that are associated with the Visitor's current session or previous 10 sessions, if any.

```
public class ExampleTracker : Sitecore.Web.UI.WebControl
    {
        protected override void DoRender(System.Web.UI.HtmlTextWriter output)
        {
            if (Tracker.CurrentVisit == null)
                return;

            if (Tracker.CurrentVisit.Keywords != null
                && !String.IsNullOrEmpty(Tracker.CurrentVisit.Keywords.Text))
            {
                output.WriteLine("Search keywords for current visit: " +
                    Tracker.CurrentVisit.Keywords.Text + ".<br/>");
                return;
            }

            const int checkVisits = 10;

            Sitecore.Analytics.Data.DataAccess.VisitorLoadOptions vOptions =
                new Sitecore.Analytics.Data.DataAccess.VisitorLoadOptions
            {
                Start = Tracker.CurrentVisit.VisitorVisitIndex - 1,
                Count = Tracker.CurrentVisit.VisitorVisitIndex - checkVisits,
                VisitLoadOptions = VisitLoadOptions.Visits
            };

            foreach (VisitorDataSet.VisitsRow visit in
                Tracker.Visitor.GetVisits(vOptions).Where(
                visit => visit.VisitId !=
                   Tracker.CurrentVisit.VisitId).OrderByDescending(
                visit => visit.VisitorVisitIndex))
            {
                if (visit.Keywords != null &&
                  !String.IsNullOrEmpty(visit.Keywords.Text))
                {
                    output.WriteLine("Last search keywords from " +
                        visit.StartDateTime + " visit: " +
                        visit.Keywords.Text + "<br/>");
                    return;
                }
            }
            output.WriteLine("No search keywords for current or last " +
                checkVisits + " visits.<br/>");
        }
```

3.1.2 How to Access GEO IP Data

The following code access Geo IP data from the current visit. In this example, the `BusinessName` property is tested to see if a specific `BusinessName` is associated with the current visit IP address.

```
        public class GeoIPTracker : Sitecore.Web.UI.WebControl
            {

                protected override void DoRender(System.Web.UI.HtmlTextWriter output)
                {
                    string ip = new IPAddress(Tracker.CurrentVisit.Ip).ToString();

                    if (Tracker.CurrentVisit == null)
                        return;

                    if (!Tracker.CurrentVisit.UpdateGeoIpData())
                        output.Write("GeoIP information not " +
                          "available within prescribed time.<br/>");

                    else if (Tracker.CurrentVisit.BusinessName == "IP_NOT_FOUND"
                        || Tracker.CurrentVisit.BusinessName == "N/A")
                        output.Write("GeoIP information not avaialble for "
                            + ip + ".<br/>");

                    else if (String.IsNullOrEmpty(Tracker.CurrentVisit.BusinessName))
                        output.Write("No business name in GeoIP data for " + ip +
                            " (error contacting provider).<br/>");

                    else
                        output.Write("Business name from GeoIP record: "
                            + Tracker.CurrentVisit.BusinessName + ".<br/>");
                }
            }
```

3.1.3 How to Access Campaign Data

The following code tests to see if there is a Campaign ID associated with the current visit.

```
        public class CampaignTracker : Sitecore.Web.UI.WebControl
            {
                protected override void DoRender(System.Web.UI.HtmlTextWriter output)
                {

                    if (Tracker.CurrentVisit.IsCampaignIdNull())
                    {
                        output.WriteLine("No campaign ID is associated " +
                            "with the current visit.<br/>");
                    }
                    else
                    {
                        output.WriteLine("Campaign for current visit: " +
                            Tracker.CurrentVisit.CampaignId.ToString() + ".<br/>");
                    }
                }
            }
```

3.1.4 How to Access Profile Data

The following code shows how to access Visitor Profile information.

```
using System.Linq;
using Sitecore.Analytics;
…

public class ProfileTracker : Sitecore.Web.UI.WebControl
{

   protected override void DoRender(System.Web.UI.HtmlTextWriter output)
   {
      if (Tracker.CurrentVisit.Profiles.Count() > 0)
      {
         output.Write("Start Profile<br/>");

         foreach (Sitecore.Analytics.Data.DataAccess.DataSets.
            VisitorDataSet.ProfilesRow row in Tracker.CurrentVisit.Profiles)

                    output.WriteLine("Data from profile " + row[0] + "<br/>");
      }
      else
      {
         output.WriteLine("No profile values.<br/>");
      }
   }
}
```

3.1.5 How to Access External User Data

The following code shows how to access the external user data in order to link visitor information with a Sitecore user account.

```
using Sitecore.Analytics;

public class ExternalUserTracker : Sitecore.Web.UI.WebControl
{
  protected override void DoRender(System.Web.UI.HtmlTextWriter output)
  {
    if (Tracker.Visitor == null)
    {
      return;
    }

    string externalUser = Tracker.Visitor.ExternalUser;
    output.Write("External User name " + externalUser + ".<br/>");

    // You can also set this property programmatically
    // if you are using a custom data provider

    Tracker.Visitor.ExternalUser = @"extranet\John";
  }
}
```

3.1.6 How to Access Tags Data

The following code shows how to access tags data in order to store and retrieve custom analytics attributes for a visitor.

```
using Sitecore.Analytics;
using Sitecore.Analytics.Data.DataAccess.DataSets;

public class TagTracker : Sitecore.Web.UI.WebControl
{
  protected override void DoRender(System.Web.UI.HtmlTextWriter output)
  {
    if (Tracker.Visitor == null)
    {
      return;
    }

    // To create the Engagement Analytics tag named 'Email'
    // with the 'test@hostname.com' value for the current visitor.
    Tracker.Visitor.Tags.Add("Email", "test@hostname.net");

    // To create the Engagement Analytics tag named 'Acquaintance'
    // with some GUID value.
    Tracker.Visitor.Tags.Add(System.Guid.NewGuid());

    // To change the Engagement Analytics tag named 'Email'
    // with a new 'test@hostname.com' value
    // or create a new one if it does not exist.
    Tracker.Visitor.Tags.Set("Email", "test2@hostname.net");

    // To get the Engagement Analytics tag named 'Email'.
    VisitorDataSet.VisitorTagsRow row = Tracker.Visitor.Tags.Find("Email");
    string tagValue = row.TagValue;
  }
}
```

Note
You do not need to commit after updating a tag.

Note
You cannot remove an analytics tag using the API.

3.2 Accessing Tracking Field Data

In order to access Visitor data in the Engagement Analytics API, you primarily use the following classes:

- Sitecore.Analytics.Data.TrackingField
- Sitecore.Analytics.Data.ContentProfile
- Sitecore.Analytics.Data.ContentProfileKeyData

Most of the following code samples require this namespace `using` directive:

```
using Sitecore.Analytics.Data;
```

3.2.1 How to Access Profile Data

The following code shows how to access the assigned Profile information for a specific item.

```
using System.Linq;
using Sitecore.Analytics.Data;
using Sitecore.Data;
using Sitecore.Data.Fields;
using Sitecore.Data.Items;
using Sitecore.Diagnostics;

...

public class Profile : Sitecore.Web.UI.WebControl
{
  protected override void DoRender(System.Web.UI.HtmlTextWriter output)
  {
    Item homeItem =
       Sitecore.Data.Database.GetDatabase("master").GetItem("/sitecore/content/Home");

    Field innerField = homeItem.Fields["__Tracking"];
    if (innerField == null)
    {
      Log.Error(string.Format("Tracking field was not found in item '{0}' ('{1}')",
                 homeItem.ID, homeItem.Paths.FullPath), this);
      output.WriteLine("No profile values.<br/>");
    }
    else
    {
      TrackingField trackingField = new TrackingField(innerField);

      ContentProfile profile = trackingField.Profiles.FirstOrDefault(profileData =>
                profileData.Name.Equals("Score") && profileData.IsSavedInField);
      output.WriteLine("Profile " + profile.Name + "<br/>");

      ContentProfileKeyData[] profileKeys = profile.Keys;
      foreach (ContentProfileKeyData profileKey in profileKeys)
      {
        output.WriteLine("Profile key name " + profileKey.Name + "<br/>");
        output.WriteLine("Profile key value " + profileKey.Value + "<br/>");
      }
    }
  }
}
```