FREE eBook

LEARNING

google-analytics-api

#google-analytics-api

Table of Contents

About

Chapter 1: Getting started with google-analytics-api

Remarks

The Google Analytics APIs allow you to access data within Google Analytics. It should not be confused with the measurement protocol which is used for inserting data into Google Analytics.

The Google Analytics API is split into serval parts.

Google Analytics Reporting APIs

1. The Google Analytics Core Reporting API v3 gives you access to most of the report data in Google Analytics. With the Core Reporting API you can:

   - Build custom dashboards to display Google Analytics data.
   - Save time by automating complex reporting tasks.
   - Integrate your Google Analytics data with other business applications.

2. The Google Analytics Reporting API V4 is the most advanced programmatic method to access report data in Google Analytics. With the Google Analytics Reporting API, you can:

   - Build custom dashboards to display Google Analytics data.
   - Automate complex reporting tasks to save time.
   - Integrate your Google Analytics data with other business applications.

2. The Real Time Reporting API enables you to request real time data—for example, real time activity on a view—for an authenticated user. You can use the Real Time Reporting API to:

   - Display active viewers of a page and create a sense of urgency for users looking at an item with finite inventory.
   - Display the most popular content such as the top 10 active pages.
   - Create and display a real time dashboard.

   The Real Time Reporting API, in limited beta, is available for developer preview only. Sign up to access the API.

3. The Multi-Channel Funnels Reporting API enables you to request Multi-Channel Funnels data for an authenticated user. Data is derived from conversion path data, which shows user interactions with various traffic sources over multiple sessions prior to converting. This allows you to analyze how multiple marketing channels influence conversions over time. For more details on what data is available, read the About Multi-Channel Funnels, as well as About Multi-Channel Funnels Data. With the Multi-Channel Funnels Reporting API you can:

   - Create custom reports using Multi-Channel Funnels data. For example, you could use the Top Conversion Paths data to report on attributes such as relative position of

    interactions in a conversion path.
- Integrate Multi-Channel Funnels data with your business data. For example, you could correlate online conversion data with offline sales data and media cost data to arrive at a more complete picture of marketing ROI.
- Display Multi-Channel Funnels in new environments. For instance, you could create visualizations and other presentations of the data that communicate the value of different marketing channels in driving conversions.

Helpers

1. The Metadata API returns the list and attributes of columns (i.e. dimensions and metrics) exposed in the Google Analytics reporting APIs. Attributes returned include UI name, description, segments support, and more. You can use the Metadata API to:

    - Automatically discover new columns.
    - Access all dimensions and metrics attributes for Google Analytics reporting APIs.

    Note: This only returns metadata for the Core Reporting API and the Reporting API. Not real-time metadata.

2. The Google Analytics Embed API is a JavaScript library that allows you to easily create and embed a dashboard on a 3rd party website in a matter of minutes. It gives you a set of pluggable components that can work together to build complex tools, making it both simple and powerful at the same time.

Configuration APIs

1. The Analytics Management API allows for programmatic access to the Google Analytics configuration data. You can build applications to more efficiently manage large or complex Analytics accounts. Large companies with many properties can automate account setup. Even if you are building a reporting application the Management API provides you tools to navigate your account. You can use the Google Analytics Management API to:

    - List all the Account, Property and View information for a user.
    - Manage Properties, Views, and Goals.
    - Manage user permissions for an account hierarchy.
    - Retrieve a View ID to use with the Core Reporting API.
    - Determine which goals are active and access their configured names.
    - Manage Links between Analytics properties and AdWords accounts.
    - Manage Remarketing Audiences.

    Write operations in the Management API (e.g. create, update, delete, patch) for Web Property, View (Profile), and Goal resources is currently available as a developer preview in **limited beta**. If you're interested in using these features, request access to the beta.

2. The Provisioning API can be used to create new Google Analytics accounts and enable Google Analytics for your customers at scale. It is intended for qualified service providers and large partners. For example, you could use the Provisioning API as part of a new user

onboarding process to create a new Google Analytics account for a client and then use additional Management API resources to programmatically configure the account and link it to AdWords. This can all be automated and initiated from within your own admin or reporting interface.

The Provisioning API is available by invitation only. We are not currently accepting new projects.

Versions

There are currently three versions of the Google Analytics API live.

Google Analytics V2 Version 2.4 of the Core Reporting API is an XML-only API that is mostly backwards compatible with version 2.3. **(Legacy)**

Google Analytics V3 which includes the Core Reporting API, Management API, MetaData API and an number of other APIs. These APIs return Json

Google Analytics V4 which currently only includes the reporting API and is considered to be the most advanced way of retrieving reporting data from Google Analytics.

Examples

Accessing Google Analytics APIs

You can technically access the Google Analytics APIs using any programing language that can handle a HTTP Post or HTTP Get request.

That being said, Google has also created a number of official standard client libraries to help you with this. Using a standard client library for your chosen programming language can be much easier than coding it from the ground up yourself.

OFFICAL Client libraries with Google Analytics API support:

1. Google APIs PHP Client library - GitHub
2. Google APIs .Net Client library - GitHub NuGet
3. Google APIs Python Client library - GitHub
4. Google APIs Java Client library - link
5. Google APIs Objective-C library - GitHub

There are more libraries here.

Introduction

The Google Analytics APIs allow you to access data within Google Analytics. It should not be confused with the measurement protocol which is used for inserting data into Google Analytics.

The Google Analytics API is split into serval parts.

Google Analytics Reporting APIs

1. The Google Analytics Core Reporting API v3 gives you access to most of the report data in Google Analytics. With the Core Reporting API you can:

   - Build custom dashboards to display Google Analytics data.
   - Save time by automating complex reporting tasks.
   - Integrate your Google Analytics data with other business applications.

2. The Google Analytics Reporting API V4 is the most advanced programmatic method to access report data in Google Analytics. With the Google Analytics Reporting API, you can:

   - Build custom dashboards to display Google Analytics data.
   - Automate complex reporting tasks to save time.
   - Integrate your Google Analytics data with other business applications.

2. The Real Time Reporting API enables you to request real time data—for example, real time activity on a view—for an authenticated user. You can use the Real Time Reporting API to:

   - Display active viewers of a page and create a sense of urgency for users looking at an item with finite inventory.
   - Display the most popular content such as the top 10 active pages.
   - Create and display a real time dashboard.

   The Real Time Reporting API, in limited beta, is available for developer preview only. Sign up to access the API.

3. The Multi-Channel Funnels Reporting API enables you to request Multi-Channel Funnels data for an authenticated user. Data is derived from conversion path data, which shows user interactions with various traffic sources over multiple sessions prior to converting. This allows you to analyze how multiple marketing channels influence conversions over time. For more details on what data is available, read the About Multi-Channel Funnels, as well as About Multi-Channel Funnels Data. With the Multi-Channel Funnels Reporting API you can:

   - Create custom reports using Multi-Channel Funnels data. For example, you could use the Top Conversion Paths data to report on attributes such as relative position of interactions in a conversion path.
   - Integrate Multi-Channel Funnels data with your business data. For example, you could correlate online conversion data with offline sales data and media cost data to arrive at a more complete picture of marketing ROI.
   - Display Multi-Channel Funnels in new environments. For instance, you could create visualizations and other presentations of the data that communicate the value of different marketing channels in driving conversions.

Helpers

1. The Metadata API returns the list and attributes of columns (i.e. dimensions and metrics) exposed in the Google Analytics reporting APIs. Attributes returned include UI name, description, segments support, and more. You can use the Metadata API to:

    - Automatically discover new columns.
    - Access all dimensions and metrics attributes for Google Analytics reporting APIs.

    Note: This only returns metadata for the Core Reporting API and the Reporting API. Not real-time metadata.

2. The Google Analytics Embed API is a JavaScript library that allows you to easily create and embed a dashboard on a 3rd party website in a matter of minutes. It gives you a set of pluggable components that can work together to build complex tools, making it both simple and powerful at the same time.

Configuration APIs

1. The Analytics Management API allows for programmatic access to the Google Analytics configuration data. You can build applications to more efficiently manage large or complex Analytics accounts. Large companies with many properties can automate account setup. Even if you are building a reporting application the Management API provides you tools to navigate your account. You can use the Google Analytics Management API to:

    - List all the Account, Property and View information for a user.
    - Manage Properties, Views, and Goals.
    - Manage user permissions for an account hierarchy.
    - Retrieve a View ID to use with the Core Reporting API.
    - Determine which goals are active and access their configured names.
    - Manage Links between Analytics properties and AdWords accounts.
    - Manage Remarketing Audiences.

    Write operations in the Management API (e.g. create, update, delete, patch) for Web Property, View (Profile), and Goal resources is currently available as a developer preview in **limited beta**. If you're interested in using these features, request access to the beta.

2. The Provisioning API can be used to create new Google Analytics accounts and enable Google Analytics for your customers at scale. It is intended for qualified service providers and large partners. For example, you could use the Provisioning API as part of a new user onboarding process to create a new Google Analytics account for a client and then use additional Management API resources to programmatically configure the account and link it to AdWords. This can all be automated and initiated from within your own admin or reporting interface.

    The Provisioning API is available by invitation only. We are not currently accepting new projects.

Hello World Reporting API - Rest

```
POST https://analyticsreporting.googleapis.com/v4/reports:batchGet?access_token={Access token
from auth request}
{
  "reportRequests":[
  {
    "viewId":"XXXX",
    "dateRanges":[
      {
        "startDate":"2015-06-15",
        "endDate":"2015-06-30"
      }],
    "metrics":[
      {
        "expression":"ga:sessions"
      }],
    "dimensions": [
      {
        "name":"ga:browser"
      }]
    }]
}
```

Read Getting started with google-analytics-api online: https://riptutorial.com/google-analytics-api/topic/7253/getting-started-with-google-analytics-api

Chapter 2: Authentication

Syntax

- GET https://accounts.google.com/o/oauth2/auth?client_id={clientid}&redirect_uri=urn:ietf:wg:oauth:2.0:oob&scope={Scopes}&response_type=code
- POST https://accounts.google.com/o/oauth2/token code={Code from above call}&client_id={ClientId}&client_secret={ClientSecret}&redirect_uri=urn:ietf:wg:oauth:2.0:oob&grant_
- POST https://accounts.google.com/o/oauth2/token client_id={ClientId}&client_secret={ClientSecret}&refresh_token={From above call}&grant_type=refresh_token

Parameters

Parameter	Description
Client Id	From Google Developer console identifies your project or application
secret	From Google Developer console identifies your project or application
redirect_uri	From Google Developer location where the authentication should be returned to. In the case of Native applications `urn:ietf:wg:oauth:2.0:oob` can be used to denote localhost

Remarks

In order to access any Google API you need to identify yourself as a developer and identify your project. We do that by creating a new project on Google Developers console.

When you create your project you if you want to access the Google Analytics APIs you must enable the APIs you intend to access.

- **Reporting API**: Access to the Google Analytics Reporting API v4.
- **Analytics API**: Access to everything else.

Now you must decide how you would like to access the data.

With Google Data there are two types of Data Public and Private.

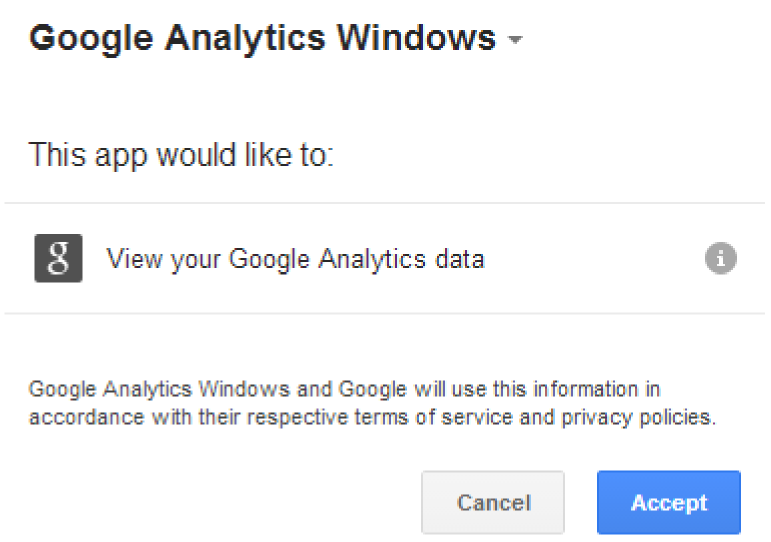
- public data is not owned by a user. The metadata API is a public API you don't need to be logged in to access that data.
- The Reporting API contains a users Google Analytics data it is private you cant look at it unless the user has given you permission to access it.

If you are only accessing public data then all you need do is create a public API key and you will be able to access the API in question. If you are going to be accessing private user data then you will need to create either Oauth2 credentials or service account credentials.

Authorization Oauth2

To access private user data we must have permission of the owner of the data to access it. Oauth2 allows us to request that access from a user.

You have probably seen before Oauth2 before.


Google Analytics Windows

This app would like to:

View your Google Analytics data

Google Analytics Windows and Google will use this information in accordance with their respective terms of service and privacy policies.

The Application "Google Analytics Windows" is requesting access to view the users "Google Analytics Data"

1. Google Analytics windows is the name of the project that was created on Google Developer console.
2. Google Analytics Data is the scope of permissions that we asked for.

Scope We need to tell the user what we intend to do the Google analytics API has two scopes that you can use.

1. https://www.googleapis.com/auth/analytics.readonly
2. https://www.googleapis.com/auth/analytics

It is best to only request the scopes that you need. If you will only be reading a users data then you only need to request the readonly scope.

Authorization service accounts

Service accounts are different in that they are pre-approved. If you create service account credentials you as the developer can take the service account email and add it as a user on your Google Analytics account **At the account level** this will grant the service account access to the data. You wont need to pop up the authentication window and request access. The service account will have access to the data for as long as it is a user on the Google Analytics account.

Conclusion

Authentication is needed to access most of the data exposed by the Google Analytics API.

> **You can not use client login / login and password to access any Google API as of May 2015. You must use Open authentication.**

**

Examples

Oauth2 C#

Example uses the Google APIs Dotnet client library.

> PM> Install-Package Google.Apis.AnalyticsReporting.v4

```
    /// <summary>
    /// This method requests Authentcation from a user using Oauth2.
    /// Credentials are stored in System.Environment.SpecialFolder.Personal
    /// Documentation https://developers.google.com/accounts/docs/OAuth2
    /// </summary>
    /// <param name="clientSecretJson">Path to the client secret json file from Google
Developers console.</param>
    /// <param name="userName">Identifying string for the user who is being
authentcated.</param>
    /// <returns>DriveService used to make requests against the Drive API</returns>
    public static AnalyticsReportingService AuthenticateOauth(string clientSecretJson, string
userName)
    {
        try
        {
            if (string.IsNullOrEmpty(userName))
                throw new Exception("userName is required.");
            if (string.IsNullOrEmpty(clientSecretJson))
                throw new Exception("clientSecretJson is required.");
            if (!File.Exists(clientSecretJson))
                throw new Exception("clientSecretJson file does not exist.");

            // These are the scopes of permissions you need.
            string[] scopes = new string[] { AnalyticsReportingService.Scope.AnalyticsReadonly
};   // View your Google Analytics Data

            UserCredential credential;
            using (var stream = new FileStream(clientSecretJson, FileMode.Open,
FileAccess.Read))
            {
                string credPath =
System.Environment.GetFolderPath(System.Environment.SpecialFolder.Personal);
                credPath = Path.Combine(credPath, ".credentials/",
System.Reflection.Assembly.GetExecutingAssembly().GetName().Name);

                // Requesting Authentication or loading previously stored authentication for
userName
                credential =
GoogleWebAuthorizationBroker.AuthorizeAsync(GoogleClientSecrets.Load(stream).Secrets,
                                                                             scopes,
                                                                             userName,
CancellationToken.None,
                                                                          new
FileDataStore(credPath, true)).Result;
            }

            // Create Drive API service.
            return new AnalyticsReportingService(new BaseClientService.Initializer()
            {
                HttpClientInitializer = credential,
                ApplicationName = string.Format("{0} Authentication",
System.Reflection.Assembly.GetExecutingAssembly().GetName().Name),
            });
        }
        catch (Exception ex)
        {
            Console.WriteLine("Create Oauth2 DriveService failed" + ex.Message);
            throw new Exception("CreateOauth2DriveFailed", ex);
        }
    }
```

Service Account Authentcation Vb.net

Sample uses Install-Package Google.Apis.AnalyticsReporting.v4

```
 Public Shared Function getServiceInitializer() As BaseClientService
    Dim serviceAccountCredentialFilePath = "Path to Json service account key file"
REM from Google Developers console
    Dim myKeyEMail = "XXX@developer.gserviceaccount.com"   REM from Google Developers console
    Dim scope =
Google.Apis.AnalyticsReporting.v4.AnalyticsReportingService.Scope.AnalyticsReadonly


    Try

        Dim credential
        Using stream As New FileStream(serviceAccountCredentialFilePath, FileMode.Open,
FileAccess.Read)

            credential = GoogleCredential.FromStream(stream).CreateScoped(scope)

        End Using

        Dim Initializer As New BaseClientService.Initializer()
        Initializer.HttpClientInitializer = credential
        Initializer.ApplicationName = "SRJCGMail"

        Dim service As New AnalyticsReportingService(Initializer)
        Return service

    Catch ex As Exception
        Console.WriteLine(ex.Message)
        Return Nothing
    End Try

End Function
```

Read Authentication online: https://riptutorial.com/google-analytics-api/topic/7268/authentication

Chapter 3: Error Responses

Syntax

- 400 invalidParameter Indicates that a request parameter has an invalid value.
- 400 badRequest Indicates that the query was invalid. E.g., parent ID was missing or the combination of dimensions or metrics requested was not valid.
- 403 insufficientPermissions Indicates that the user does not have sufficient permissions for the entity specified in the query.
- 403 dailyLimitExceeded Indicates that user has exceeded the daily quota (either per project or per view (profile)).
- 403 usageLimits.userRateLimitExceededUnreg Indicates that the application needs to be registered in the Google API Console.
- 403 userRateLimitExceeded Indicates that the user rate limit has been exceeded. The maximum rate limit is 10 qps per IP address.
- 403 rateLimitExceeded Indicates that the global or overall project rate limits have been exceeded. Retry using exponential back-off. You need to slow down the rate at which you are sending the requests.
- 403 quotaExceeded Indicates that the 10 concurrent requests per view (profile) in the Core Reporting API has been reached.
- 429 RESOURCE_EXHAUSTED AnalyticsDefaultGroupCLIENT_PROJECT-1d Indicates that the Requests per day per project quota has been exhausted.
- 429 RESOURCE_EXHAUSTED AnalyticsDefaultGroupCLIENT_PROJECT-100s Indicates that the Requests per 100 seconds per project quota has been exhausted.
- 429 RESOURCE_EXHAUSTED AnalyticsDefaultGroupUSER-100s Indicates that the requests per 100 seconds per user per project quota has been exhausted.
- 429 RESOURCE_EXHAUSTED DiscoveryGroupCLIENT_PROJECT-100s Indicates that the discovery requests per 100 seconds quota has been exhausted.
- 500 internalServerError Unexpected internal server error occurred. Do not retry this query more than once.
- 503 backendError Server returned an error. Do not retry this query more than once.

Parameters

Name	Description
domain	Location of the error ex: global
reason	Reason for the error
message	A message explaining the error and a possible solution.
locationType	The location type of the error ex: paramater
location	The actual location of the error

Remarks

If an request is successful, the API returns a 200 HTTP status code along with the requested data in the body of the response.

If an error occurs with a request, the API returns an HTTP status code and reason in the response based on the type of error. Additionally, the body of the response contains a detailed description of what caused the error.

Examples

400 invalidParameter

It is important to read the error response that is returned by the Google Analytics API server. In a lot of cases they can tell you exactly what is wrong.

```
400 invalidParameter

{
 "error": {
  "errors": [
   {
    "domain": "global",
    "reason": "invalidParameter",
    "message": "Invalid value '-1' for max-results. Value must be within the range: [1, 
1000]",
    "locationType": "parameter",
    "location": "max-results"
   }
  ],
  "code": 400,
  "message": "Invalid value '-1' for max-results. Value must be within the range: [1, 1000]"
 }
}
```

In this case the message:

> "message": "Invalid value '-1' for max-results. Value must be within the range: [1, 1000]",

Is telling us that we sent a -1 value for the parameter max-results which is not valid we can only send a value from 1-1000.

Read Error Responses online: https://riptutorial.com/google-analytics-api/topic/7376/error-responses

Chapter 4: Metadata api

Syntax

- HTTP GET https://www.googleapis.com/analytics/v3/metadata/{reportType}/columns?key={APIKey}
- HTTP GET https://www.googleapis.com/analytics/v3/metadata/{reportType}/columns?access_token={Access_token}

Parameters

Parameter name	Description
reportType	Report type. Allowed Values: **ga**. Where **ga** corresponds to the Core Reporting API.

Remarks

The Metadata API returns the list and attributes of columns (i.e. dimensions and metrics) exposed in the Google Analytics reporting APIs (v2,v3 and v4). Attributes returned include UI name, description, segments support, and more.

You can use the Metadata API to:

- Automatically discover new columns
- Access all dimensions and metrics attributes for Google Analytics reporting APIs

This is the same list as is in the Dimensions & Metrics Explorer.

> Note: Real-time and Multi-Channel Funnels dimensions and metrics are not currently available.

Examples

Rest Example

Calls to the Metadata API are with HTTP Get:

Using Public API Key

> GET https://www.googleapis.com/analytics/v3/metadata/ga/columns?key={YOUR_API_KEY}

Using Access token from either Oauth2 or Service account authentication

GET https://www.googleapis.com/analytics/v3/metadata/ga/columns?access_token={Authentcated_Access_Token}

Java example

uses the Java Client library

```
/**
 * 1. Execute a Metadata Request
 * An application can request columns data by calling the list method on the Analytics service
object.
 * The method requires an reportType parameter that specifies the column data to retrieve.
 * For example, the following code requests columns for the ga report type.
 */

try {
  Columns results = getMetadata(analytics);
  // Success

} catch (GoogleJsonResponseException e) {
  // Catch API specific errors.
  handleApiError(e);
} catch (IOException e) {
  // Catch general parsing network errors.
  e.printStackTrace();
}

/**
 * 2. Print out the Columns data
 * The components of the result can be printed out as follows:
 */

private static Columns getMetadata(Analytics analytics) throws IOException {
  String reportType = "ga";
  return analytics.metadata()
      .columns()
      .list(reportType)
      .execute();
}


private static void printMetadataReport(Columns results) {
  System.out.println("Metadata Response Report");
  printReportInfo(results);
  printAttributes(results.getAttributeNames());
  printColumns(results.getItems());
}


private static void printReportInfo(Columns results) {
  System.out.println("## Metadata Report Info ##");
  System.out.println("Kind: " + results.getKind());
  System.out.println("Etag: " + results.getEtag());
  System.out.println("Total Results: " + results.getTotalResults());
  System.out.println();
}
```

```
private static void printAttributes(List<String> attributeNames) {
  System.out.println("## Attribute Names ##");
  for (String attribute : attributeNames) {
    System.out.println(attribute);
  }
}


private static void printColumns(List<Column> columns) {
  System.out.println("## Columns ##");

  for (Column column : columns) {
    System.out.println();
    System.out.println("Column ID: " + column.getId());
    System.out.println("Kind: " + column.getKind());

    Map<String, String> columnAttributes = column.getAttributes();

    for (Map.Entry<String, String> attribute: columnAttributes.entrySet()) {
      System.out.println(attribute.getKey() + ": " + attribute.getValue());
    }
  }
}
```

Note: first version copied from Metadata.list

PHP Example

Uses the PHP client library

```
/**
 * 1. Execute a Metadata Request
 * An application can request columns data by calling the list method on the Analytics service
object.
 * The method requires an reportType parameter that specifies the column data to retrieve.
 * For example, the following code requests columns for the ga report type.
 */

try {

  $results = $analytics->metadata_columns->listMetadataColumns('ga');
  // Success

} catch (apiServiceException $e) {
  // Handle API service exceptions.
  $error = $e->getMessage();
}


/**
 * 2. Print out the Columns data
 * The components of the result can be printed out as follows:
 */

function printMetadataReport($results) {
  print '<h1>Metadata Report</h1>';
  printReportInfo($results);
  printAttributes($results);
  printColumns($results);
}


function printReportInfo(&$results) {
  $html = '<h2>Report Info</h2>';
  $html .= <<<HTML
<pre>
Kind                  = {$results->getKind()}
Etag                  = {$results->getEtag()}
Total Results         = {$results->getTotalResults()}
</pre>
HTML;
  print $html;
}


function printAttributes(&$results) {
  $html = '<h2>Attribute Names</h2><ul>';
  $attributes = $results->getAttributeNames();
  foreach ($attributes as $attribute) {
    $html .= '<li>'. $attribute . '</li>';
  }
  $html .= '</ul>';
  print $html;
}


function printColumns(&$results) {
  $columns = $results->getItems();
  if (count($columns) > 0) {
    $html = '<h2>Columns</h2>';
    foreach ($columns as $column) {
      $html .= '<h3>' . $column->getId() . '</h3>';
      $column_attributes = $column->getAttributes();
      foreach ($column_attributes as $name=>$value) {
        $html .= <<<HTML
<pre>
{$name}: {$value}
</pre>
HTML;
      }
    }
  } else {
    $html = '<p>No Results Found.</p>';
  }
  print $html;
}
```

Note: original version copied from metadata.list

Python Example

Uses the Python client library

```
# 1. Execute a Metadata Request
# An application can request columns data by calling the list method on the Analytics service
object.
# The method requires an reportType parameter that specifies the column data to retrieve.
# For example, the following code requests columns for the ga report type.

try:
  results = service.metadata().columns().list(reportType='ga').execute()

except TypeError, error:
  # Handle errors in constructing a query.
  print ('There was an error in constructing your query : %s' % error)

except HttpError, error:
  # Handle API errors.
  print ('Arg, there was an API error : %s : %s' %
         (error.resp.status, error._get_reason()))

# 2. Print out the Columns data
# The components of the result can be printed out as follows:

def print_metadata_report(results):
  print 'Metadata Response Report'
  print_report_info(results)
  print_attributes(results.get('attributeNames'))
  print_columns(results)


def print_report_info(columns):
  print "Metadata Report Info"
  if columns:
    print 'Kind           = %s' % columns.get('kind')
    print 'Etag           = %s' % columns.get('etag')
    print 'Total Results  = %s' % columns.get('totalResults')


def print_attributes(attributes):
  if attributes:
    print 'Attribute Names:'
    for attribute in attributes:
      print attribute

def print_columns(columns_data):
  if columns_data:
    print 'Columns:'

    columns = columns_data.get('items', [])

    for column in columns:
      print
      print '%15s = %35s' % ('Column ID', column.get('id'))
      print '%15s = %35s' % ('Kind', column.get('kind'))

      column_attributes = column.get('attributes', [])

      for name, value in column_attributes.iteritems():
        print '%15s = %35s' % (name, value)
```

Note: original version copied from metadata.list

C# example

Uses the .Net Client library

PM> Install-Package Google.Apis.Analytics.v3

```
var metadataService = new AnalyticsMetaDataService(new BaseClientService.Initializer()
            {
                ApiKey = {Public API KEY},
                ApplicationName = "Metadata api",
            });

var result = Service.Metadata.Columns.List("ga").Execute();
```

Read Metadata api online: https://riptutorial.com/google-analytics-api/topic/7297/metadata-api

Chapter 5: Real-time API

Syntax

- GET https://www.googleapis.com/analytics/v3/data/realtime?ids=[Analytics view Id]&metrics=[Real-time metrics]&access_token=[Access_token_from_Authentcation]
- GET https://www.googleapis.com/analytics/v3/data/realtime?ids=[Analytics view Id]&metrics=[Real-time metrics]&dimensions=[Real-time metrics]&access_token=[Access_token_from_Authentcation]

Parameters

(Required) Parameter name	Description
ids	Unique table ID for retrieving Analytics data. Table ID is of the form ga:XXXX, where XXXX is the Analytics view (profile) ID.
metrics	A comma-separated list of Analytics metrics. E.g., 'rt:activeUsers'. At least one metric must be specified.
(Optional) Parameter name	**Description**
dimensions	A comma-separated list of real-time dimensions. E.g., 'rt:medium,rt:city'.
filters	A comma-separated list of dimension or metric filters to be applied to real-time data.
max-results	The maximum number of entries to include in this feed.
sort	A comma-separated list of dimensions or metrics that determine the sort order for real-time data.
(Standard) Parameter name	**Description**
callback	Name of the JavaScript callback function that handles the response. Used in JavaScript JSON-P requests.
prettyPrint	Returns the response in a human-readable format if true. Default value: true. When this is false, it can reduce the response payload size, which might lead to better performance in some environments.

(Required) Parameter name	Description
quotaUser	Lets you enforce per-user quotas from a server-side application even in cases when the user's IP address is unknown. This can occur, for example, with applications that run cron jobs on App Engine on a user's behalf. You can choose any arbitrary string that uniquely identifies a user, but it is limited to 40 characters.Overrides userIp if both are provided. Learn more about capping usage.
userIp	Lets you enforce per-user quotas when calling the API from a server-side application. Learn more about capping usage.

Remarks

> **The Real Time Reporting API, in limited beta, is available for developer preview only. Sign up to access the API.**

The Real Time Reporting API enables you to request real time data—for example, real time activity on your property—for an authenticated user.

You can use the Real Time Reporting API to:

- Display active viewers of a page and create a sense of urgency for users looking at an item with finite inventory.
- Display the most popular content such as the top 10 active pages.
- Create and display a real time dashboard.

Authorization

Calls to the Google Analytics Real-time API requires authorization with at least one of the following scopes (read more about authentication and authorization).

Scope	access granted
https://www.googleapis.com/auth/analytics	Full access to Google Analytics accounts up to the level of the authenticated users access.
https://www.googleapis.com/auth/analytics.readonly	Read only access to the Authenticated users Google Analytics accounts

Examples

PHP Example

Uses the PHP client library

```
/**
 * 1.Create and Execute a Real Time Report
 * An application can request real-time data by calling the get method on the Analytics
service object.
 * The method requires an ids parameter which specifies from which view (profile) to retrieve
data.
 * For example, the following code requests real-time data for view (profile) ID 56789.
 */
$optParams = array(
    'dimensions' => 'rt:medium');

try {
  $results = $analytics->data_realtime->get(
      'ga:56789',
      'rt:activeUsers',
      $optParams);
  // Success.
} catch (apiServiceException $e) {
  // Handle API service exceptions.
  $error = $e->getMessage();
}


/**
 * 2. Print out the Real-Time Data
 * The components of the report can be printed out as follows:
 */

function printRealtimeReport($results) {
  printReportInfo($results);
  printQueryInfo($results);
  printProfileInfo($results);
  printColumnHeaders($results);
  printDataTable($results);
  printTotalsForAllResults($results);
}

function printDataTable(&$results) {
  if (count($results->getRows()) > 0) {
    $table .= '<table>';

    // Print headers.
    $table .= '<tr>';

    foreach ($results->getColumnHeaders() as $header) {
      $table .= '<th>' . $header->name . '</th>';
    }
    $table .= '</tr>';

    // Print table rows.
    foreach ($results->getRows() as $row) {
      $table .= '<tr>';
        foreach ($row as $cell) {
          $table .= '<td>'
                 . htmlspecialchars($cell, ENT_NOQUOTES)
                 . '</td>';
        }
      $table .= '</tr>';
    }
    $table .= '</table>';

  } else {
    $table .= '<p>No Results Found.</p>';
  }
  print $table;
}

function printColumnHeaders(&$results) {
  $html = '';
  $headers = $results->getColumnHeaders();

  foreach ($headers as $header) {
    $html .= <<<HTML
<pre>
Column Name       = {$header->getName()}
Column Type       = {$header->getColumnType()}
Column Data Type  = {$header->getDataType()}
</pre>
HTML;
  }
  print $html;
}

function printQueryInfo(&$results) {
  $query = $results->getQuery();
  $html = <<<HTML
<pre>
Ids         = {$query->getIds()}
Metrics     = {$query->getMetrics()}
Dimensions  = {$query->getDimensions()}
Sort        = {$query->getSort()}
Filters     = {$query->getFilters()}
Max Results = {$query->getMax_results()}
</pre>
HTML;

  print $html;
}

function printProfileInfo(&$results) {
  $profileInfo = $results->getProfileInfo();

  $html = <<<HTML
<pre>
Account ID               = {$profileInfo->getAccountId()}
Web Property ID          = {$profileInfo->getWebPropertyId()}
Internal Web Property ID = {$profileInfo->getInternalWebPropertyId()}
Profile ID               = {$profileInfo->getProfileId()}
Profile Name             = {$profileInfo->getProfileName()}
Table ID                 = {$profileInfo->getTableId()}
</pre>
HTML;

  print $html;
}

function printReportInfo(&$results) {
  $html = <<<HTML
  <pre>
Kind                  = {$results->getKind()}
```

```
ID                      = {$results->getId()}
Self Link               = {$results->getSelfLink()}
Total Results           = {$results->getTotalResults()}
</pre>
HTML;

  print $html;
}

function printTotalsForAllResults(&$results) {
  $totals = $results->getTotalsForAllResults();

  foreach ($totals as $metricName => $metricTotal) {
    $html .= "Metric Name  = $metricName\n";
    $html .= "Metric Total = $metricTotal";
  }

  print $html;
}
```

Original version coped from official documentation

Read Real-time API online: https://riptutorial.com/google-analytics-api/topic/7305/real-time-api

Chapter 6: Reporting API (Analytics v4)

Remarks

The Google Analytics Reporting API V4 is the most advanced programmatic method to access report data in Google Analytics. With the Google Analytics Reporting API, you can:

- Build custom dashboards to display Google Analytics data.
- Automate complex reporting tasks to save time.
- Integrate your Google Analytics data with other business applications.

Features

Google Analytics is built upon a powerful data reporting infrastructure. The Google Analytics Reporting API V4 gives you access to the power of the Google Analytics platform. The API provides these key features:

- Metric expressions: The API allows you to request not only built-in metrics but also combination of metrics expressed in mathematical operations. For example, you can use the expression ga:goal1completions/ga:sessions to request the goal completions per number of sessions.
- Multiple date ranges: The API allows you in a single request to get data in two date ranges.
- Cohorts and Lifetime value: The API has a rich vocabulary to request Cohort and Lifetime value reports.
- Multiple segments: The API enables you to get multiple segments in a single request.

Examples

Single report Example using Oauth2 C#

This example uses the official Google .net Client library.

> PM> Install-Package Google.Apis.AnalyticsReporting.v4

Authorization Requires one of the following OAuth scopes:

- https://www.googleapis.com/auth/analytics.readonly
- https://www.googleapis.com/auth/analytics

Oauth2

```
// These are the scopes of permissions you need.
string[] scopes = new string[] { AnalyticsReportingService.Scope.AnalyticsReadonly };   //
View your Google Analytics Data

UserCredential credential;
using (var stream = new FileStream(clientSecretJson, FileMode.Open, FileAccess.Read))
```

```
{
    string credPath =
System.Environment.GetFolderPath(System.Environment.SpecialFolder.Personal);
    credPath = Path.Combine(credPath, ".credentials/",
System.Reflection.Assembly.GetExecutingAssembly().GetName().Name);

    // Requesting Authentication or loading previously stored authentication for userName
   credential =
GoogleWebAuthorizationBroker.AuthorizeAsync(GoogleClientSecrets.Load(stream).Secrets,
       scopes,
      userName,
      CancellationToken.None,
      new FileDataStore(credPath, true)).Result;
}

// Create Reporting API service.
var service = new AnalyticsReportingService(new BaseClientService.Initializer()
            {
             HttpClientInitializer = credential,
             ApplicationName = string.Format("{0} Authentication",
System.Reflection.Assembly.GetExecutingAssembly().GetName().Name),
            });
```

Reporting Request

```
// Create the DateRange object.
DateRange June2015 = new DateRange() { StartDate = "2015-01-01", EndDate = "2015-06-30" };
DateRange June2016 = new DateRange() { StartDate = "2016-01-01", EndDate = "2016-06-30" };
List<DateRange> dateRanges = new List<DateRange>() { June2016, June2015 };


// Create the ReportRequest object.
// This should have a large number of rows
ReportRequest reportRequest = new ReportRequest
{
    ViewId = ConfigurationManager.AppSettings["GoogleAnaltyicsViewId"],
    DateRanges = dateRanges,
    Dimensions = new List<Dimension>() { new Dimension() { Name = "ga:date" }, new Dimension()
{ Name = "ga:usertype" } },
    Metrics = new List<Metric>() { new Metric() { Expression= "ga:users" }, new Metric() {
Expression = "ga:sessions" } },
    PageSize = 1000,
};

List<ReportRequest> requests = new List<ReportRequest>();
requests.Add(reportRequest);


var getReport = new GetReportsRequest() { ReportRequests = requests };
var response = service.Reports.BatchGet(getReport).Execute();
```

Single Report Example Rest

API requests are HTTP POST with the access token attached at the end of the API end point.

Authorization Requires one of the following OAuth scopes:

- https://www.googleapis.com/auth/analytics.readonly

- https://www.googleapis.com/auth/analytics

Note when posting the data use `ContentType = "application/Json";`

```
https://analyticsreporting.googleapis.com/v4/reports:batchGet?Access_token={from auth}
{
  "reportRequests":[
  {
    "viewId":"XXXX",
    "dateRanges":[
      {
        "startDate":"2015-06-15",
        "endDate":"2015-06-30"
      }],
    "metrics":[
      {
        "expression":"ga:sessions"
      }],
    "dimensions": [
      {
        "name":"ga:browser"
      }]
    }]
}
```

Read Reporting API (Analytics v4) online: https://riptutorial.com/google-analytics-api/topic/7265/reporting-api--analytics-v4-

Credits

S. No	Chapters	Contributors
1	Getting started with google-analytics-api	Community, DaImTo, K48, MarkeD
2	Authentication	DaImTo, Matt
3	Error Responses	DaImTo
4	Metadata api	DaImTo
5	Real-time API	DaImTo
6	Reporting API (Analytics v4)	DaImTo