

MailChimp

The module OrbitOne.MailChimp

MailChimp has over 15,000 customers, manages over 65,000 opt-in lists containing over 75 million subscribers, and we deliver millions of emails a day. We strive to make MailChimp easy and affordable enough for a small business to get started, but powerful enough for a large company that's looking for an enterprise level solution.

<http://www.mailchimp.com>

The MailChimp API is a way for you to "sync" your customer database, CRM, CMS, or e-commerce shopping cart with MailChimp. It's really powerful stuff.

<http://www.mailchimp.com/api>

|  |  |
| --- | --- |
| Date: | 27 November 2008 |
| Developed by: | Wim De Coninck |
| Reference: |  |
| Developed for: | Enter the client or “Orbit One Internal” here |

Contents

[1. Goal 3](#_Toc223412769)

[2. Process diagram 4](#_Toc223412770)

[2.1. Overview 4](#_Toc223412771)

[2.2. Login 5](#_Toc223412772)

[2.3. Get MailChimp list members 6](#_Toc223412773)

[2.4. Compare list members and IMailChimpContacts 7](#_Toc223412774)

[3. How to use the module 8](#_Toc223412775)

[3.1. The IMailChimpContact interface 8](#_Toc223412776)

[3.2. The update method 8](#_Toc223412777)

[4. Configuration 9](#_Toc223412778)

# Goal

The goal of this module is to facilitate a synchronisation with a list in MailChimp (<http://www.mailchimp.com>), in such a way that a piece of client software does not need to know anything about the MailChimp API or its way of working. The module exposes 1 method: Update. This method receives a list of IMailChimpContacts (which need to be implemented by the client) and results in a list of GUID’s. The resulting GUID’s are the list members in MailChimp that have been put on opt-out (either through the web-based user interface or through the unsubscribe link at the bottom of a recipients’ email) in this update cycle.

# Process diagram

## Overview



## Login



## Get MailChimp list members



## Compare list members and IMailChimpContacts

**2**

**1**



1. Deletes:
   1. When a contact is opt-out and the contact is present in the MailChimp list. (this is necessary to be able to reopt-in through the datasource (CRM, SQL-Server, SharePoint, …))
   2. MailChimp list members that do not exist in the contacts (based on the id’s)
   3. MailChimp list members that opted out
   4. Opt-in contacts that are present in the MailChimp list members (based on the id’s) but that do not have the same email address (these need to be deleted because MailChimp uses the email address as their unique key/id)
2. Creates:
   1. Contact that is opt-in and does not exist in the MailChimp list members
   2. Contact that is opt-in and that is opt-in in the MailChimp list members (comparison based on the id) and that has a different email
   3. Updates: the also translate to creates (because the methods are the same in the MailChimp API). A update is triggered when a field (except email) differs in the MailChimp list member or the contact.

# How to use the module

## The IMailChimpContact interface

The interface IMailChimpContact says that the implementor needs to have a certain set of properties:

1. ExternalId (Guid): this is to have a means to identify the record on both MailChimp and DataSource
2. FieldValues (IDictionary<string,string>): here the key is the field name (as specified in the configuration) and the value is the value of that column in the MailChimp list.
3. Subscribed (bool): to indicate wether a contact is subscribed to a list or not.
4. Email (string): the email address of the contact
5. Groups (List<string>): this can be empty or can contain a list of groups that will be translated to interest groups in MailChimp. The interest groups are used by MailChimp to facilitate segmentation in MailChimp.

## The update method

The update method gets a list of IMailChimpContacts and returns the opt-out MailChimp list members’ guids.

A small example of a possible implementation:

try

{

var chimp = new MailChimp();

var result = chimp.Update(contacts);

foreach (var guid in result)

{

Console.WriteLine(guid.ToString());

}

Console.ReadLine();

}

catch (Exception e)

{

Console.Out.WriteLine(e);

Console.ReadLine();

}

# Configuration

The client side of this module (your program if you are using this module) should provide the following in the app.config:

<configSections>

<section name="MailChimpConfiguration"

type="OrbitOne.MailChimp.MailChimpConfiguration,

OrbitOne.MailChimp"/>

</configSections>

<MailChimpConfiguration

APIKey="yourAPIkey"

UserName="username"

Password="p455w0rd"

ListName="MyList"

RetryCount="3">

<Columns>

<add name="First Name" tag="FNAME" required="false" />

<add name="Last Name" tag="LNAME" required="false" />

</Columns>

</MailChimpConfiguration>

* APIKey: This is for the following version of the .NET Wrapper. This is coming out in a few weeks.
* UserName: specify your MailChimp login name
* Password: specify your MailChimp password
* ListName: the name of the list you want to synchronize with.
* RetryCount: how many times should the login be retried (when the MailChimp site has a lot of visitors at any given time, it is possible that the login will fail the first time due to a timeout. I would recommend to keep the retrycount at a minimum of 2)
* Columns: In the columns you need to specify the name, the tag and the requirement. You can add as many columns as you want.
  + name: the name of the column (as displayed in the web interface)
  + tag: the program friendly name (no spaces, all upper case, not too long, unique). **Do not use Id, ID, INTEREST as a tag. Those will result in problems on MailChimps side.**
  + required: sets a field as required. If no data is provided in the IMailChimpContact for a field, there won’t be any errors. But the contact will not be fully processed (it will not be added to the list) and you will get a message of that in the report.