Integración con el Bigmailer.cloud - API

Registros en la Red, S.L.

www.registros.net

www.bigmailer.net

Tabla de contenido

1 Introducción

Bigmailer.cloud como plataforma de envió masivo correos y comunicación a gran escala con los clientes ha sido preparado para fácil integración con existentes sistemas informáticos incluyendo las paginas web.

Como medio de la integración se esta utilizando dos estandartes comunes: SOAP y llamadas http con el uso de codificación de entrada y salida con JSON.

2 Seguridad

Para poder acceder al API de la plataforma Bigmailer.cloud es necesario utilizar la clave API. Puedes conocer tus claves API dentro del panel de control de bigmailer.cloud, apartado *Gestión* → *Integración* → *Las herramientas de desarrolladores.*

Existen dos tipos de calves API. Clave publica permite efectuar algunas operaciones que no ponen en peligro el sistema ni la información. Esta clave se puede publicar con el texto abierto ya que no permite acceder a información delicada ni hacer operaciones que requieren autorización. Clave publica mas bien se utiliza para identificar tu cuenta en la plataforma Bigmailer.cloud. En cambio, las claves privadas permiten efectuar varias operaciones y entre ellas extracción o manipulación de datos.

Es importante que protejas tus clave API. No la facilites a los terceros ni utilices en conexiones no seguras. Las claves API se deben utilizar solo bajo el protocolo https.

3 Integración Web

La plataforma Bigmailer.cloud permite fácil integración con las paginas web. Esta opción puede ser muy interesante para vincular el formulario de alta en el newsletter (boletín). La clase o web service *Contact* puede trabajar en el modo de trabajo *newsletter* lo permite una integración rápida de boletín. El sistema bigmailer se ocupara del envío de un correos de confirmación de alta almacenando toda la información requerida por la nueva normativa RGPD.

Para garantizar maxima seguridad es altamente importante incluir un sistema CAPTCHA dentro del formulario del alta del newsletter.

4 SOAP

El sistema bigmailer ofrece un explorador SOAP que contiene la lista de servicios y métodos disponible y toda la información necesaria para la integración.

Para utilizar el explorador, visite el sitio: https://bigmailer.cloud/soap

Para agaragar contactos nuevos es altamente aconsejable utilizar el servicio *Contact* que puede encontrar siguiendo el vinculo: https://bigmailer.cloud/es/soap/index/class/Api-Soap-Integration-Contact

El servicio *Contact* permite aprovechar todas las ventajas de la herramienta *Integración* que se puede encontrar en el panel de control de bigmailer.cloud en la sección *Gestión*. Las funcionalidades destacadas de la herramienta *Integración* son:

- Selección automática de la base de datos;
- Registro de acciones.

Para conocer modos de trabajo de servicio o clase *Contact* consulte el subtitulo *Contact – Modos de trabajo*.

5 Entrada JSON

Entrada JSON puede ser una atractiva alternativa para el estándar de SOAP. Este método permite la integración incluso en los sistemas que no están preparados para uso de SOAP.

Para utilizar la entrada, lo único que tienes que hacer es efectuar una llamada http a la dirección URL de web service del sistema bigmailer.cloud.

La URL del web service: https://bigmailer.cloud/api/json/run

5.1 Entrada

La entrada tomo como datos la información pasada dentro del parámetro *input*. El parámetro puede pasarse tanto con el método POST como GET pero es importante garantizar estructura y codificación adecuada de la información.

5.1.1 Estructura

La información de la entrada debe estructurarse como una *array asociativa* que contiene siguientes elementos importantes como claves:

- *class*
- *method*
- *arguments*

class – Contiene el nombre de la clase que se va a utilizar en la llamada. Los nombres de clases corresponden a los nombres que aparecen en el explorador SOAP. Por ejemplo para utilizar la clase *Contact* hay que pasar como el el nombre completo: *\Api\Soap\Integration\Contact*

method – El nombre de método a cual se hace la llamada. Los nombres corresponden a los nombres que aparecen en el explorador SOAP. Por ejemplo para añadir el contacto según la IP del visitante de la web hay que utilizar el método *addByIP*.

arguments – El parámetro arguments debe contener una array con los parámetros que se pasan al método, de acuerdo con la información que aparece en el explorador SOAP. Por ejemplo para el método *addByIP* los parámetros requeridos son: *key, email, name, ip y mode*.

Un ejemplo de la estructura de la entrada codificada en PHP se puede encontrar en el listado 1.

```
$input = array(
        'class' => '\Api\Soap\Integration\Contact',
        'method' => 'addByIP',
        'arguments' => array(
                '77b1fb0f84c424b2ba6de76b1ffce172766b0834',
                'test@test.net',
                'Test',
                '8.8.8.8',
                0,
                ),
);
```

Listado 1: Estructura de la entrada

5.1.2 Codificación

El parámetro input tiene que estar codificado para poder ser pasado mediante el protocolo http. Para codificar el paramento hay que efectuar dos operaciones: 1) Serializar a formato JSON y 2) codificación según la normativa RFC 3986.

En el listado 2 se presenta el proceso de codificación PHP utilizando las funciones *json_encode()* y *rawurlencode()*.

```
$input = json_encode($input);
$input = rawurlencode($input);
```

Listado 2: Codificación de la entrada

5.2 El resultado de la llamada

Como el resultado de la llamada el sistema devuelve información codificada como JSON. La información después de deserializar tiene estructura de una array asociativa con al menos dos campos: *code* y *result*.

El campo *code* indica si la llamada se ha efectuado con éxito (código 200) o ha fallado (código 5xx).

Si el código de la respuesta es 200, el campo *result* contiene el resultado de la ejecución del metodo indicado en la entrada.

```
$input = array(
	'class' => '\Api\Soap\Integration\Contact',
	'method' => 'addByIP',
	'arguments' => array(
		'77b1fb0f84c424b2ba6de76b1ffce172766b0834',
		'test@test.net',
		'Test',
		'8.8.8.8',
		0,
		),
);

$input = json_encode($input);
$input = rawurlencode($input);

$ch = curl_init();
curl_setopt($ch, CURLOPT_URL,"https://bigmailer.cloud/api/json/run");
curl_setopt($ch, CURLOPT_POST, TRUE);
curl_setopt($ch, CURLOPT_POSTFIELDS, "input={$input}");
curl_setopt($ch, CURLOPT_RETURNTRANSFER, true);
$response = curl_exec($ch);
curl_close ($ch);

$response = json_decode($response, TRUE);

var_dump($response);
```

Listado 3: Implementacion de la llamada JSON

El listado 3 contiene un código PHP completo para efectuar una llamada JSON.

6 Contact – Modos de trabajo

La clase o web service *Contact* es la mejor opción para insertar de contactos nuevos mediante el API de Bigmailer.cloud. El funcionamiento de este método se puede controlar desde panel de bigmailer accediendo a la herramienta *Integración* dentro de la sección *Gestión*. Ademas la herramienta facilita un registro detallado de las interacciones.

La clase ofrece tres métodos *state*(), a*ddByIP()* y *addByCountry().*

El método *state()* permite obtener información acerca del estado de un contacto. El método toma solo un parámetro tipo *string* que debe de ser una dirección email correcta. Los posibles resultados del método se ha presentado en la tabla 1.

El método a*ddByIP()* permite asignación automática de la base de datos donde se va a añadir el contacto nuevo. Como referencia para elegir la base de datos se toma información de geolocalización basada en la IP.

El método a*ddByCountry()* permite asignación automática de la base de datos donde se va a añadir el contacto nuevo. Como referencia para elegir la base de datos se toma el código del país según la normativa ISO 3166-1.

Los parámetros que toman los dos métodos han sido descritas en las tablas 2 y 3.

Valor	**Descripción**
0	El contacto no existe.
1	El contacto existe pero no tiene asignados los permisos para mailing.
2	El contacto existe pero el envío esta prohibido.
3	El contacto existe y tiene permiso para el mailing.
4	El contacto existe y tiene permiso para el mailing pero no se cumple con la normativa RGPD.
5	El contacto existe y tiene permiso para el mailing pero el control de RGPD has sido desactivado. Se supone que la información requerida por la normativa RGPD se guarda fuera del bigmailer.cloud.

Tabla 1: Resultados del método state()

Parámetro	Tipo	Descripción
key	string	API Key
email	string	Dirección de correo.
name	string	Nombre de contacto. Se puede dejar vacío.
ip	string	IP del contacto.
mode	integer	El modo de trabajo.

Tabla 2: Parámetros del método addByIP()

Parámetro	Tipo	Descripción
key	string	API Key
email	string	Dirección de correo.
name	string	Nombre de contacto. Se puede dejar vacío.
country	string	Código del país según la normativa ISO 3166-1
ip	string	IP del contacto. Dependiendo del modo del trabajo puede ser obligatorio.
mode	integer	El modo de trabajo.

Tabla 3: Parámetros del método addByCountry()

6.1 El modo de trabajo

El modo tel trabajo indica al sistema de que manera se tiene que tratar el contacto nuevo. En la tabla 4 se describe con detalle todos modos de trabajo soportados por métodos *addByIP()* y *addByCountry()*.

Valor	Descripción
0	El contacto se va a importar sin permiso para el envio.
1	El contacto se va a importar en modo newsletter. Sin permiso para el envío pero al contacto se va a enviar un correo con la solicitud de la confirmación de alta. Si el contacto acepta la alta, el sistema va a asignar el permiso para envío y va a guardar toda la información requerida por RGPD.
2	El contacto se va a importar sin permiso pero al contacto se le va a enviar una solicitud de permiso para el envío de mailing. Si el contacto acepta la alta, el sistema va a asignar el permiso para envío y va a guardar toda la información requerida por RGPD.
3	El contacto se va a importar con permiso para envío. Ademas se va a tomar la IP y la fecha y hora actual para guardar el registros requerido por RGPD. En este modo la IP es requerida también si se utiliza el método *addByCountry()*.
4	El contacto se va a importar con permisos pero el sistema va a desactivar control de RGPD para esta dirección. En este modo de trabajo se supone que la información requerida por RGPD se guarda fuera del bigmailer.cloud.

Tabla 4: Modos del trabajo

6.2 Integración PHP

En caso de la integración con sistemas escritos en PHP están disponibles las librerías que facilitan de forma importante la programación.

Las librerías junto con los ejemplos se puede descargar desde :
https://github.com/registros/integration

El listado 4 presenta la interacciona con el uso de las librerías.

```
require_once 'vendor/Registros/loader.php';

use Registros\Bigmailer\Api\Contact;

$contact = new Contact('77b1fb0f84c424b2ba6de76b1ffce172766b0834');
$result = $contact->addByIP('test@test.net', 'Test', '8.8.8.8', Contact::ADD_MODE_WITH_PERMISSION);

var_dump($result);
```

Listado 4: Integración utilizando las librerías

7 Newsletter – formulario remoto

La plataforma Bigmailer.cloud permite implementación dentro del sitio web un formulario remoto implementado como un *iframe*. Esta técnica permite una rápida y cómoda implementación de un newsletter (boletín) corporativo que ademas de todas sus funcionalidades, cumple perfectamente con la normativa RGPD. Los contactos que se dan de alta mediante el formulario remoto de newsletter automáticamente reciben un email con la solicitud de confirmación de alta. El contenido de este correo se puede configurar en el apartado *Gestión* → *Correos de sistema* → *Solicitud de confirmación*.

El formulario se puede implementar como un simple código HTML. Un ejemplo de la implementación se puede encontrar en el listado 5.

```
<iframe
src='https://bigmailer.cloud/es/integration/newsletter/form/9369e4a05bf1a739aa91d682fca8c905bb7af03b'
frameborder='0' scrolling='no' style='width: 450px; height: 300px;' ></iframe>
```

Listado 5: Codigo HTML de iframe

La ultima parte de la dirección URL del atributo *src*, debe de ser la clave publica. Mas información sobre la clave publica puede encontrar en el capitulo Seguridad.

Ademas dispone de un sencillo generador de código HTML que puede encontrar en apartado *Gestión* → *Integración* → *Las herramientas de desarrolladores.*

En la tabla 5 se puede encontrar los parámetros que permiten personalizar mejor la presentación del formulario. Los parámetros de una URL deben estar codificados según la normativa RFC 3986.

Parámetro	**Descripción**	**Ejemplo**
title	Permite personalizar el titulo del formulario. Advertencia. El titulo personalizado no se traduce a otros idiomas de forma automática en el entorno internacional	https://bigmailer.cloud/es/integration/newsletter/form/9369e4a05bf1a739aa91d682fca8c905bb7af03b?title=Nuestro%20bulletin
css	La dirección URL completa a un archivos de estilos CSS.	https://bigmailer.cloud/es/integration/newsletter/form/9369e4a05bf1a739aa91d682fca8c905bb7af03b?css=https%3A%2F%2Fmi-web.net%2Fstyle.css
callback	La dirección URL completa a un script para informar la sitio web que el cliente ya se ha registrado.	https://bigmailer.cloud/es/integration/newsletter/form/9369e4a05bf1a739aa91d682fca8c905bb7af03b?css=https%3A%2F%2Fmi-web.net%2Fcallback.php

Tabla 5: Parametros del formulario remoto

7.1 Integración PHP

En caso de la integración con sistemas escritos en PHP están disponibles las librerías que facilitan de forma importante la programación.

Las librerías junto con los ejemplos se puede descargar desde :
https://github.com/registros/integration

La clase *Registros\Bigmailer\Newsletter\Form* permite creación automatizada del código HTML del *iframe* del formulario remoto. Ademas la clase puede controlar si el formulario se debe presentar o no en caso de los visitantes que ya se han dado de alta en el newsletter.

La tabla 6 presenta los mas importantes métodos publicados por la clase *Registros\Bigmailer\Newsletter\Form.*

Método	**Argumentos**	**Descripción**
__construct	String $public_key	El constructor como único argumento toma la clave publica de la cuenta del Bigmailer.cloud.
setTitle	String $title	Permite personalizar el titulo del formulario remoto. Advertencia. El titulo personalizado no se traduce a otros idiomas de forma automática en el entorno internacional
setCss	String $css	Permite añadir un archivos de estilos CSS. La ruta puede ser absoluta o relativa en cual caso, el sistema interpreta como la base la raíz (*root*) del sitio web.
setCallback	String $callback	Permite añadir un script que se va a llamar en caso de alta en el newsletter. La ruta puede ser absoluta o relativa en cual caso, el sistema interpreta como la base la raíz (*root*) del sitio web.
setWidth	String $width	Permite fijar una anchura según la sintaxis CSS.
setHeight	String $height	Permite fijar una altura según la sintaxis CSS.
setCallbackController	ICallback $callback_controller	Permite poner un gestor de *callback* propia o personalizado.
isRegistered		Devuelve TRUE si el visitante de la sesiona actual ya se ha registrado. Requiere llamada callback.
render	Boolean $show_registered = FALSE	Devuelve el código HTML listo para el uso dentro del documento HTML. El parámetro $show_registered permite decidir si el formularios se presentara en caso de los visitantes que ya se han registrado.

Tabla 6: Métodos de la clase Form

Las librerías de integración proporcionan clase Registros\Bigmailer\Newsletter\Callback que perite controlar si el cliente de la sesión actual ya se ha registrado en el newsletter.

La tabla 7 presenta los mas importantes métodos publicados por la clase *Registros\Bigmailer\Newsletter*\Callback.

Método	Argumentos	Descripción
setStorageMode	Int $storage_mode	Decide de que forma el gestor va a guardad la información sobre los visitantes. Posibles valores: Callback::STORAGE_FILE – la información se guarda en los archivos. Este modo no requiere el inicio de la sesiona. Callback::STORAGE_SESSION – requiere el inicio de la sesiona pero no los permisos de escritura en la raíz de la web. El sistema por defecto utiliza el almacenamiento en los archivos para evitar el inicio de la sesión innecesariamente.
getSession		Devuelve ID de la sesiona. Advertencia, en el modo de almacenamiento en los archivos la ID de la sesiona puede ser diferente que el resultado de la funciona *session_id()*.
IsRegistered		Devuelve TRUE si el visitante de la sesiona actual ya se ha registrado. Requiere llamada callback.
register	String $session	Se utiliza para registrar un visitante cuando se da de alta en el newsletter. Se debe llamar dentro del script de callback. Como argumento toma la ID de la session generada durante la generación del la URL al formulario remoto.

Table 7: Métodos de la clase Callback

En los listados 6 y 7se presente scripts de generación del formulario y pala la llamada callback. Ejemplos funcionales se puede encontrar el las librerías de integración, en la carpeta *examples*.

```
require_once 'vendor/Registros/loader.php';

use Registros\Bigmailer\Newsletter\Form;

$newsletter = new Form('9369e4a05bf1a739aa91d682fca8c905bb7af03b');
$newsletter->setCss('style.css');
$newsletter->setCallback('callback.php');

echo $newsletter->render();
```

Listado 6: Generación del formulario remoto (index.php)

```
require_once 'vendor/Registros/loader.php';

use Registros\Bigmailer\Newsletter\Callback;

$callback = new Callback();
$callback->register(@$_GET['session']);
```

Listado 7: Scripts para las llamadas callback (callback.php)

8 Soporte técnico

En caso de dudas o problemas con la integración, las preguntas se puede enviar por el correo electrónica a la dirección de *soporte@registros.net* .