



# Dokumentacja API v2.0

Kraków 22 listopada 2011

FreshMail, ul. Fabryczna 20a, 31-553 Kraków  
tel. +48 12 617 61 40, info@freshmail.pl, freshmail.pl



**Wersja dokumentu:** 2.0.4

**Autor dokumentu:** Michał Białas <[michal.bialas@freshmail.pl](mailto:michal.bialas@freshmail.pl)>

## Zmiany

- 2.0.4 – dodano nową metodę putSubscribers oraz przykład jej wykorzystania, zmiana url do wsdl'a oraz soap'a
- 2.0.3 – pierwsza opublikowana wersja

## Opis API

Api systemu FreshMail zostało zrealizowane za pomocą technologii SOAP 1.1. Umożliwia logowanie, zarządzanie kontami, listami, subskrybentami, polami dodatkowymi oraz wysyłanie kampanii.

Adres pliku wsdl: <https://api.freshmail.pl/soap?wsdl>

## Obsługa błędów

Każda metoda dostępna w API zwraca wyjątek w przypadku wystąpienia błędu.

## Opis metod

### Metody Ogólne

#### 1. loginAccount

Logowanie do konta. Metoda zwraca id sesji użytkownika.

```
string loginAccount(string login, string password)
```

<b>login</b>	login, email użytkownika
<b>password</b>	hasło użytkownika

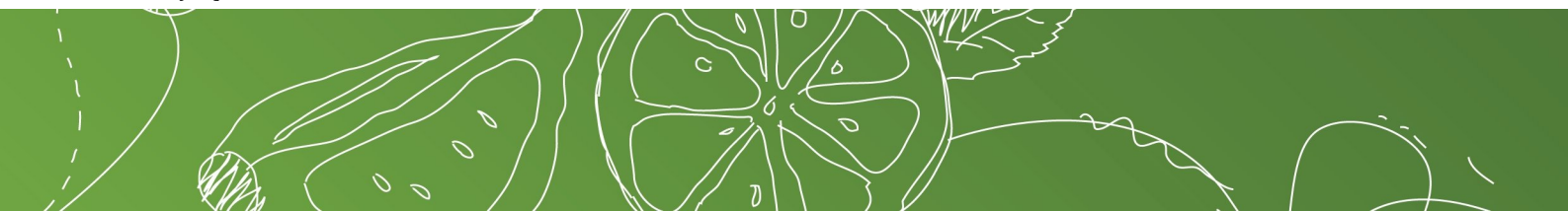
#### 2. logoutAccount

Wylogowanie użytkownika.

```
string logoutAccount()
```

*Przykład:*

```
<?php
$login = "freshmail@example.com";
$pass = "examplepassword";
$wsdlUrl = "https://api.freshmail.pl/soap?wsdl";
try {
```



```

$soap = new SoapClient($wsdlUrl);
$soap->loginAccount(array("login" => $login, "password" => $pass));
//jestem zalogowany
$soap->logoutAccount();
//zostałem wylogowany
} catch (Exception $e) {
    echo "Wystąpił błąd: " . $e->getMessage() . PHP_EOL;
}

```

## Metody do zarządzania kontami

### 1. createAccount

Tworzenie nowego konta w systemie. Metoda zwraca identyfikator użytkownika.

```

ClientAccountHash createAccount(string login, string password, string name,
                                string surname, string company, string phone,
                                bool needActivation, bool sendActivationMail)

```

<b>login</b>	login konta (adres email)
<b>password</b>	hasło
<b>name</b>	imię głównego użytkownika
<b>surname</b>	nazwisko głównego użytkownika
<b>company</b>	nazwa firmy
<b>phone</b>	numer telefonu
<b>needActivation</b>	czy wymagana jest aktywacja konta poprzez mail
<b>sendActivationMail</b>	czy wysyłać maila aktywacyjnego

### 2. updateAccount

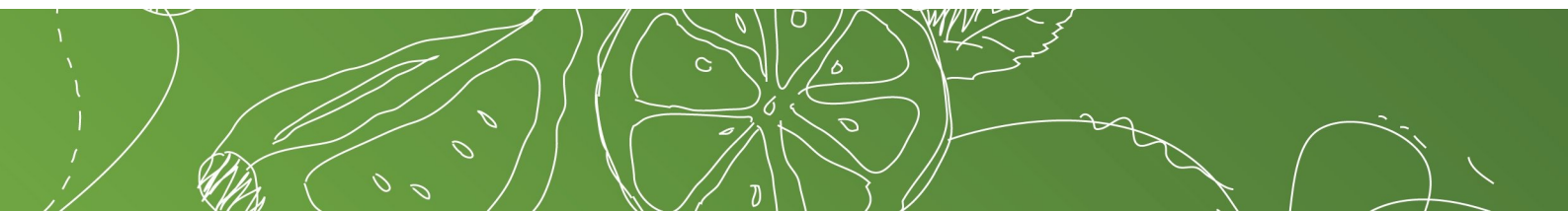
Aktualizacja bieżącego konta. Metoda zwraca 1 jeżeli wszystko się powiodło, 0 w przeciwnym wypadku.

```

integer updateAccount(string password, string name, string surname,
                      string company, string phone)

```

<b>password</b>	hasło
<b>name</b>	imię głównego użytkownika
<b>surname</b>	nazwisko głównego użytkownika
<b>company</b>	nazwa firmy
<b>phone</b>	numer telefonu



### 3. activateAccount

Aktywacja konta. Metoda zwraca 1 jeżeli wszystko się powiodło, 0 w przeciwnym wypadku.

```
integer activateAccount(string clientAccountHash)
```

<b>clientAccountHash</b>	identyfikator konta
--------------------------	---------------------

### 4. getClientHash

Pobieranie bieżącego identyfikatora konta. Zwraca identyfikator konta

```
clientAccountHash getClientHash()
```

### 5. createAccountUser

Tworzenie użytkownika do konta. Metoda zwraca identyfikator użytkownika.

```
string createAccountUser(string login, string password, string name,  
string surname, string company, string phone,  
bool needActivation, bool sendActivationMail)
```

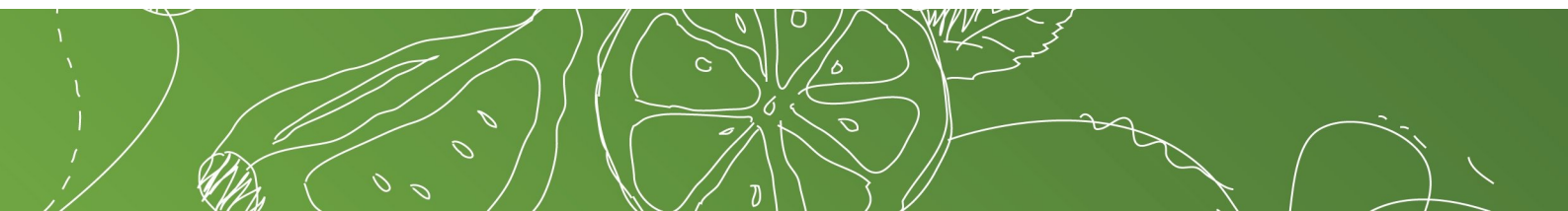
<b>login</b>	login konta (adres email)
<b>password</b>	hasło
<b>name</b>	imię głównego użytkownika
<b>surname</b>	nazwisko głównego użytkownika
<b>company</b>	nazwa firmy
<b>phone</b>	numer telefonu
<b>needActivation</b>	czy wymagana jest aktywacja konta poprzez mail
<b>sendActivationMail</b>	czy wysyłać maila aktywacyjnego

### 6. updateAccountUser

Aktualizacja użytkownika do konta. Metoda zwraca 1 jeżeli wszystko się powiodło, 0 w przeciwnym wypadku.

```
integer updateAccountUser(string password, string name, string surname,  
string company, string phone)
```

<b>password</b>	hasło
<b>name</b>	imię głównego użytkownika
<b>surname</b>	nazwisko głównego użytkownika
<b>company</b>	nazwa firmy
<b>phone</b>	numer telefonu



## 7. activateAccountUser

Aktywacja użytkownika do konta. Metoda zwraca 1 jeżeli wszystko się powiodło, 0 w przeciwnym wypadku.

```
integer activateAccountUser(string clientAccountUserHash)
```

<b>clientAccountUserHash</b>	identyfikator użytkownika
------------------------------	---------------------------

## Metody do zarządzania listami subskrypcyjnymi

### 1. createSubscriberList

Tworzenie nowej listy subskrypcyjnej. Zwraca identyfikator listy subskrypcyjnej.

```
string createSubscriberList(string name, string description)
```

<b>name</b>	nazwa listy
<b>description</b>	opis listy

### 2. updateSubscriberList

Aktualizacja danych listy subskrypcyjnej. Metoda zwraca 1 jeżeli wszystko się powiodło, 0 w przeciwnym wypadku.

```
integer updateSubscriberList(string subscriberListHash, string name,  
string description)
```

<b>subscriberListHash</b>	Identyfikator listy subskrypcyjnej
<b>name</b>	nazwa listy
<b>description</b>	opis listy

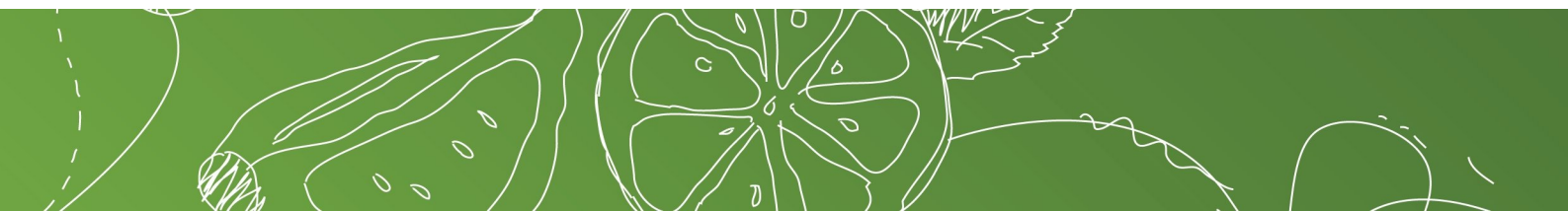
### 3. getSubscriberLists

Pobranie wszystkich list subskrypcyjnych aktualnego użytkownika zalogowanego na konto. Zwraca tablicę obiektów **subscriberList**.

```
subscriberLists getSubscriberLists()
```

obiekt **subscriberlist**

string <b>subscriberListHash</b>	identyfikator listy subskrypcyjnej
string <b>name</b>	nazwa listy
string <b>description</b>	opis listy
string <b>creation_date</b>	data utworzenia listy



## 4. createCustomField

Tworzenie nowego pola dodatkowego na liście subskrypcyjnej. Zwraca identyfikator pola dodatkowego.

```
string createCustomField(string subscriberListHash, string name,  
                        string personalizationTag, string type)
```

<b>subscriberListHash</b>	identyfikator pola dodatkowego
<b>name</b>	nazwa pola dodatkowego
<b>personalizationTag</b>	tag personalizacji
<b>type</b>	text albo number, oznacza typ pola dodatkowego

## 5. updateCustomField

Aktualizacja pola dodatkowego. Metoda zwraca 1 jeżeli wszystko się powiodło, 0 w przeciwnym wypadku.

```
integer updateCustomField(string subscriberListHash, string customFieldHash,  
                        string name, string personalizationTag)
```

<b>subscriberListHash</b>	identyfikator listy subskrypcyjnej
<b>customFieldHash</b>	identyfikator pola dodatkowego
<b>name</b>	nazwa pola dodatkowego
<b>personalizationTag</b>	tag personalizacji

## 6. getCustomFieldsFromSubscriberList

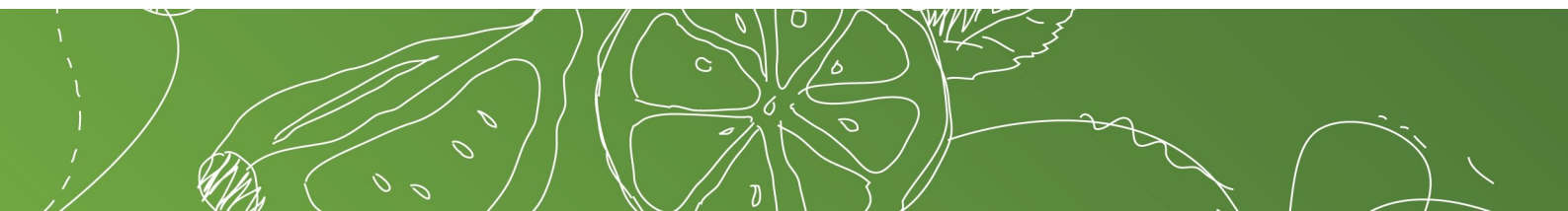
Pobranie wszystkich pól dodatkowych z listy subskrypcyjnej. Zwraca tablicę obiektów **customField**.

```
customFields getCustomFieldsFromSubscriberList(string subscriberListHash)
```

<b>subscriberListHash</b>	identyfikator listy subskrypcyjnej
---------------------------	------------------------------------

obiekt **customField**

string <b>customFieldHash</b>	identyfikator pola dodatkowego
string <b>name</b>	nazwa pola dodatkowego
string <b>personalizationTag</b>	tag personalizacji
string <b>type</b>	text albo number, oznacza typ pola dodatkowego



## Metody do zarządzania subskrybentami oraz polami dodatkowymi

### 1. createSubscriber

Tworzenie nowego subskrybenta. Metoda zwraca identyfikator subskrybenta.

```
string createSubscriber(string subscriberListHash, string email, string name,  
                        bool needActivation, bool sendActivationMail)
```

<b>subscriberListHash</b>	identyfikator listy subskrypcyjnej
<b>email</b>	email subskrybenta
<b>name</b>	nazwa subskrybenta
<b>needActivation</b>	czy wymagana jest aktywacja konta poprzez mail
<b>sendActivationMail</b>	czy wysyłać maila aktywacyjnego

### 2. updateSubscriber

Aktualizacja danych subskrybenta. Metoda zwraca 1 jeżeli wszystko się powiodło, 0 w przeciwnym wypadku.

```
integer updateSubscriber(ChooseSubscriber chooseSubscriber, Set set)
```

<b>chooseSubscriber</b>	obiekt <b>ChooseSubscriber</b> (wybór subskrybenta do modyfikacji)
<b>set</b>	obiekt <b>Set</b> (modyfikacja subskrybenta)

obiekt **ChooseSubscriber**

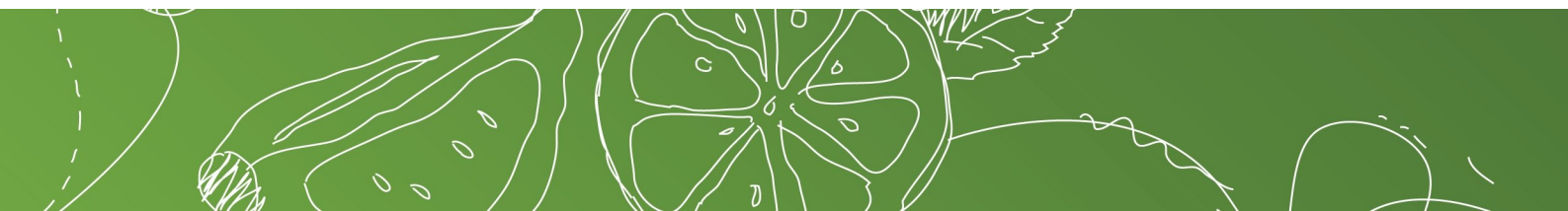
string <b>byColumn</b>	możliwe 2 wartości <code>subscriberHash</code> lub <code>subscriberEmail</code> .
string <b>columnValue</b>	w zależności od wyboru kolumny albo identyfikator subskrybenta albo jego adres email
string <b>subscriberListHash</b>	identyfikator listy subskrypcyjnej (argument opcjonalny)

obiekt **Set**

string <b>email</b>	adres email subskrybenta
string <b>name</b>	nazwa subskrybenta
string <b>state</b>	status subskrybenta. Należy użyć jednego z poniższych: <b>dontChange</b> , <b>active</b> , <b>toActivation</b> , <b>bounceSoft</b> , <b>bounceHard</b> , <b>unsubscribed</b> , <b>deleted</b> ; co znaczy odpowiednio: „nie zmieniaj”, „aktywny”, „do aktywacji”, „odbicie miękkie”, „odbicie twarde”, „wypisany”, „usunięty”
<b>customFieldsValues</b>	Tablica obiektów <b>customFieldsValue</b>

### 3. activateSubscriber

Aktywacja subskrybenta. Metoda zwraca 1 jeżeli wszystko się powiodło, 0 w przeciwnym



wypadku.

```
integer activateSubscriber(string subscriberListHash, string email)
```

<b>subscriberListHash</b>	identyfikator listy subskrypcyjnej
<b>email</b>	adres email subskrybenta

#### 4. getSubscriberByEmail

Pobranie danych subskrybenta przy pomocy adresu email. Zwraca tablicę obiektów **subscriber**.

```
subscribers getSubscriberByEmail(string email, string subscriberListHash)
```

<b>email</b>	adres email szukanego subskrybenta
<b>subscriberListHash</b>	identyfikator listy subskrypcyjnej (argument opcjonalny)

obiekt **subscriber**

string <b>subscriberListHash</b>	identyfikator listy subskrypcyjnej
string <b>subscriberHash</b>	identyfikator subskrybenta
string <b>email</b>	adres email subskrybenta
string <b>name</b>	nazwa subskrybenta
string <b>added_on</b>	data dodania subskrybenta
string <b>state</b>	status subskrybenta
string <b>last_update</b>	ostatnia aktualizacja danych subskrybenta
string <b>last_send</b>	data ostatniej wysyłki do subskrybenta
<b>customField[] customFields</b>	tablica obiektów customField

#### 5. getSubscriberByHash

Pobranie danych subskrybenta przy pomocy identyfikatora subskrybenta. Zwraca tablicę obiektów **subscriber**.

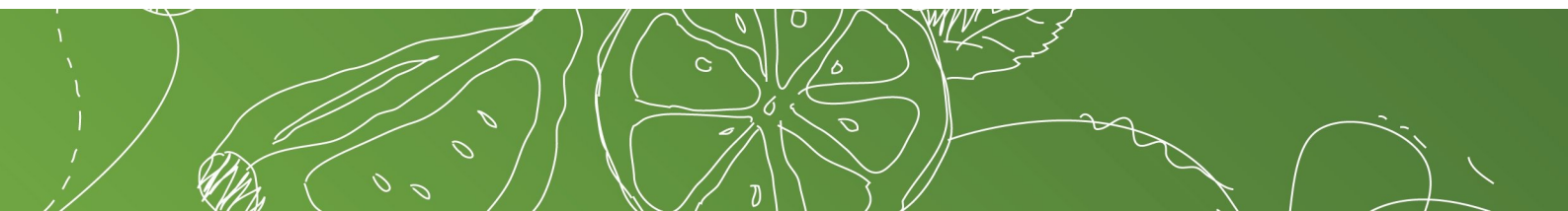
```
subscribers getSubscriberByHash(string subscriberHash)
```

<b>subscriberHash</b>	identyfikator subskrybenta
-----------------------	----------------------------

#### 6. updateCustomFieldsValues

Aktualizacja pól dodatkowych. Metoda zwraca 1 jeżeli wszystko się powiodło, 0 w przeciwnym wypadku.

```
integer updateCustomFieldsValues(UpdateCustomFieldValue[] updateCustomFieldValues)
```



<b>updateCustomFieldValues</b>	tablica obiektów UpdateCustomFieldValue
--------------------------------	---

obiekt **UpdateCustomFieldValue**

string <b>customFieldHash</b>	identyfikator pola dodatkowego
string <b>subscriberHash</b>	identyfikator subskrybenta
string <b>value</b>	wartość pola dodatkowego

## 7. **getCustomFieldsValuesForSubscriber**

Pobranie pól dodatkowych dla subskrybenta. Zwraca tablicę obiektów **customFieldsValue**.

```
customFieldsValues getCustomFieldsValuesForSubscriber(string subscriberHash)
```

<b>subscriberHash</b>	identyfikator subskrybenta
-----------------------	----------------------------

obiekt **customFieldsValue**

string <b>customFieldHash</b>	identyfikator pola dodatkowego
string <b>personalizationTag</b>	tag personalizacji
string <b>name</b>	nazwa pola dodatkowego
string <b>value</b>	wartość pola dodatkowego

## ***Metody do wysyłania kampanii***

### **1. createCampaign**

Tworzenie nowej kampanii o określonej nazwie na podstawie przesłanego tekstu i html lub przy pomocy pobierania kampanii z url'a. Metoda zwraca hash kampanii potrzebny do dalszego manipulowania ustawieniami oraz do wysyłki.

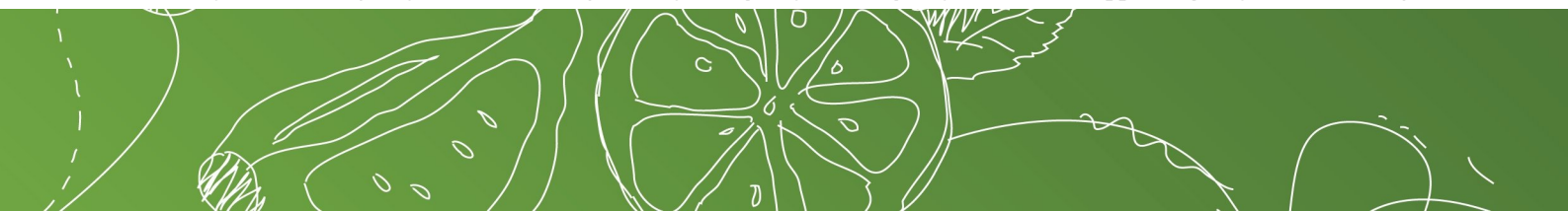
```
string createCampaign(string name, string html, string text, string url)
```

<b>name</b>	nazwa kampanii
<b>html</b>	treść html kampanii, jeżeli parametr jest pusty to jest tworzona tylko wersja tekstowa.
<b>text</b>	treść tekstowa kampanii, jeżeli parametr jest pusty to jest tworzona tylko wersja htmlowa.
<b>url</b>	odnośnik z którego zostanie pobrana kampania. Jeżeli nie jest pusty albo nie podany to zastępuje wersję html'ową kampanii. (Parametr opcjonalny).

### **2. setCampaignParameters**

Ustawienie podstawowych parametrów kampanii takich jak: temat, nadawca czy pole „odpowiedz do”. Metoda zwraca 1 jeżeli wszystko się powiodło, 0 w przeciwnym wypadku.

```
integer setCampaignParameters(string campaign, CampaignParameter[] campaignParameter)
```



<b>campaign</b>	identyfikator kampanii
<b>campaignParameter</b>	Tablica obiektów <b>CampaignParameter</b>

obiekt **CampaignParameter**

<b>string key</b>	klucz parametrów
<b>string value</b>	wartości parametrów

**Lista parametrów**

<b>subject</b>	temat mailingu
<b>fromEmail</b>	nadawca maila, adres email
<b>fromName</b>	nadawca maila, nazwa
<b>replyTo</b>	adres email „odpowiedz do”

### 3. setCampaignSubscribersLists

Ustawienie list subskrypcyjnych do których jest wysyłana kampania. Metoda zwraca 1 jeżeli wszystko się powiodło, 0 w przeciwnym wypadku.

```
integer setCampaignSubscribersLists(string campaign, SubscribersList[] subscribersLists)
```

<b>campaign</b>	identyfikator kampanii
<b>subscribersLists</b>	Tablica obiektów <b>SubscribersList</b>

obiekt **SubscribersList**

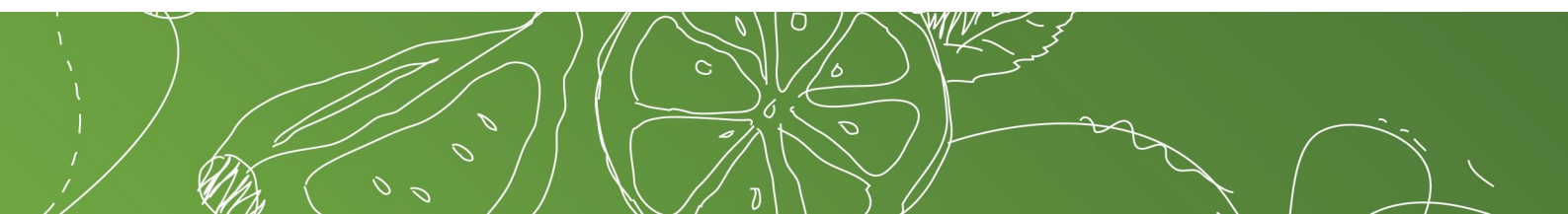
<b>string subscribersListHash</b>	identyfikator listy subskrypcyjnej, opcjonalnie można przekazać identyfikator listy subskrypcyjnej i po przecinku identyfikator grupy
-----------------------------------	---

### 4. sendTestCampaign

Wysłanie testowej kampanii. Metoda zwraca 1 jeżeli wszystko się powiodło, 0 w przeciwnym wypadku.

```
integer sendTestCampaign(string campaign, string emails, string customFieldsFill)
```

<b>campaign</b>	identyfikator kampanii
<b>emails</b>	adresy email na które zostanie wysłana kampania. Jeżeli jest więcej niż jeden to powinny być oddzielone przecinkami
<b>customFieldsFill</b>	sposób wypełniania pól dodatkowych w mailingu testowym. Na razie jedynym obsługiwanym jest <b>random</b> (wybiera losowego subskrybenta z listy subskrypcyjnej wybranych do wysłania i używa jego pól dodatkowych do personalizacji mailingu).



## 5. sendCampaign

Wysłanie kampanii. Metoda zwraca 1 jeżeli wszystko się powiodło, 0 w przeciwnym wypadku.

```
integer sendCampaign(string campaign, datetime sendTime)
```

<b>campaign</b>	identyfikator kampanii
<b>sendTime</b>	data wysłanie kampanii (argument opcjonalny), jeżeli data nie jest określona to kampania jest ustawiona do wysłania natychmiast

## 6. getCampaignState

Pobranie statusu kampanii. Zwraca liczbowy status kampanii.

```
int getCampaignState(string campaign)
```

<b>campaign</b>	identyfikator kampanii
-----------------	------------------------

Statusy kampanii

<b>1</b>	Kampania ze zdefiniowaną treścią maila
<b>2</b>	Kampania z ustawionymi parametrami
<b>3</b>	Kampania z ustawionymi listami subskrypcyjnymi
<b>4</b>	Kampania ustawiona do wysyłki
<b>5</b>	Kampania wysłana
<b>6</b>	Kampania w trakcie wysyłki

## 7. putSubscribers

Utworzenie tymczasowej listy subskrypcyjnej na potrzeby wysłania kampanii. Metoda może być wywołana wiele razy dla jednej kampanii. Każde wywołanie zwraca ten sam zestaw identyfikatorów. Metoda zwraca identyfikator listy i identyfikator grupy tymczasowej rozdzielony przecinkiem, służy jako parametr wejściowy dla metody **setCampaignSubscribersLists** jako lista subskrypcyjna.

```
string putSubscribers(string campaign, subscribersArray subscribers)
```

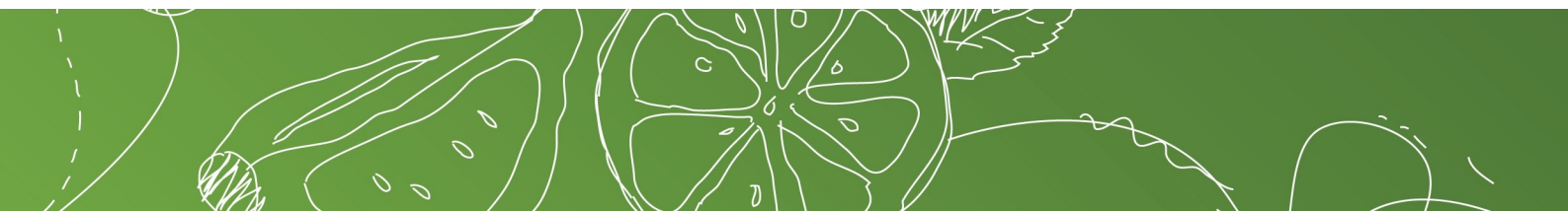
<b>campaign</b>	identyfikator kampanii
<b>subscribers</b>	obiekt zawierający tablicę subskrybentów wraz z polami dodatkowymi

Obiekt subscribersArray

<b>subscribers</b>	tablica obiektów <b>quickSubscriber</b>
--------------------	---

Obiekt **quickSubscriber**

email	adres email
name	nazwa



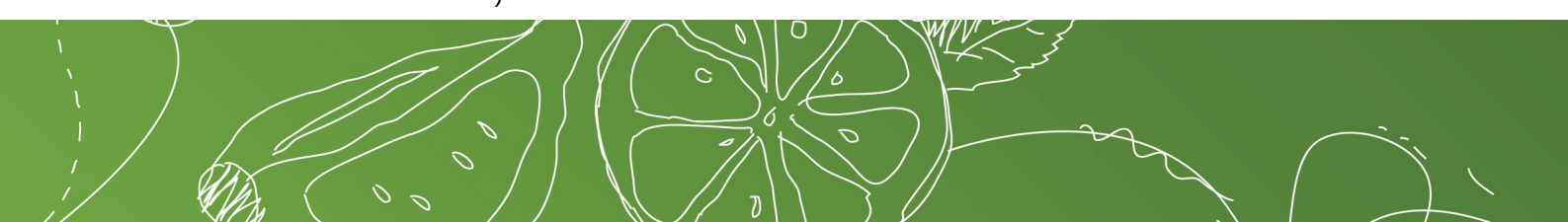
customFields	tablica obiektów <b>quickCustomFieldsValues</b>
--------------	---

### Obiekt **quickCustomFieldsValues**

personalizationTag	nazwa tagu personalizacji, nazwa pola dodatkowego
value	wartość pola dodatkowego

*Przykład:*

```
<?php
$login = "freshmail@example.com";
$pass = "examplepassword";
$wsdlUrl = "https://apo.freshmail.pl/soap?wsdl";
try {
    $soap = new SoapClient($wsdlUrl);
    $soap->loginAccount(array("login" => $login, "password" => $pass));
    $result = $soap->createCampaign(array(
        "name" => "Campaign name",
        "html" => '<p>Witaj $$imie$$</p>',
        "text" => "",
    ));
    $campaignHash = $result->return;
    $result = $soap->setCampaignParameters(array(
        "campaign" => $campaignHash,
        "campaignParameter" => array(
            array("key" => "subject", "value" => "Example subject"),
            array("key" => "fromEmail", "value" => "from@example.com"),
            array("key" => "fromName", "value" => "Example Name"),
            array("key" => "replyTo", "value" => "replyto@example.com"),
        )
    ));
    $result = $soap->putSubscribers(
        array(
            "campaign" => $campaignHash,
            "subscribers" => array(
                array(
                    "email" => "test2@example.com",
                    "name" => "Michał",
                    "customFields" => array(
                        array(
                            'personalizationTag' => 'imie',
                            'value' => 'Michał'
                        )
                    )
                )
            )
        )
    );
}
```



```
        ),
    )
)
);
$subscribersList = array(array("subscribersListHash" => $result->return));
$result = $soap->setCampaignSubscribersLists(array(
    "campaign" => $campaignHash,
    "subscribersLists" => $subscribersList
));
$result = $soap->sendCampaign(array("campaign" => $campaignHash);
$soap->logoutAccount();
} catch (Exception $e) {
    echo "Wystąpił błąd: " . $e->getMessage() . PHP_EOL;
}
```

