

Bronto API Help 2018.25.2

Contents

API Intro.....	3
TLS Support.....	3
Use Bronto's API To Manage Consent Records.....	4
SOAP API.....	5
Login To The SOAP API.....	5
SOAP Add Functions.....	12
SOAP Update Functions.....	83
SOAP Delete Functions.....	101
SOAP Misc Functions.....	108
SOAP Read Functions.....	116
Results of Add, Update, and Delete Functions.....	164
SOAP Filter Objects.....	165
SOAP Settings Objects.....	182
SOAP General Objects.....	201
SOAP Data Formats.....	249
REST API.....	249
Getting Started With REST.....	250
REST Order Service.....	250
REST Product Service.....	300
REST Campaign Service.....	316
API Errors.....	322

API Intro

Bronto's API is designed for developers who would like to interact with the Bronto application in a programmatic way. The API exposes nearly all of the capabilities presented in the graphical user interface.

The API was originally designed using the SOAP API protocol, a standard method for web applications to interoperate. Bronto's newest offerings leverage the convenience and flexibility of REST. With our REST API client, you can access and work with your product and order data.

On this site, you will find complete documentation of the SOAP and REST API calls, as well as sample code in a variety of languages.

With the Bronto API you can:

- Synchronize contacts on a regular basis
- Create, edit, and manage lists of contacts using external data stores
- Read delivery metrics and subscriber activities for use in CRM systems
- Deliver transactional and triggered messages from external systems
- Record customer conversions whether from email or other sources
- Create and update contact field data
- Import a product data feed (REST)
- Update a product (REST)
- Get the data associated with a product (REST)
- Add a cart (REST)
- Get, update, or delete a cart (REST)
- Find orders associated with a customer (REST)
- Add a new order (REST)

REST vs SOAP

When Bronto began the development of our API, we built it on the current industry standard SOAP (Simple Object Access Protocol). Within the last year our newest services have begun offering REST (REpresentational State Transfer) APIs. While there are no immediate plans to move our current SOAP APIs to REST, any future services will provide REST APIs.

So how do you know which API you need to use? If you are working with Product or Cart data, you will need to use REST calls. When you are working with Order data you will also want to use REST API calls, though you can add, update, or delete orders using SOAP functions. Everything else – contacts, lists, deliveries, messages, webforms, workflows, etc – is based on SOAP.

TLS Support

If you have developed an application that uses the Bronto API on a server, that server uses TLS when connecting to the Bronto application.

Transport Layer Security (TLS) is a security protocol that was first established in 1999 as a more secure replacement for SSL. TLS is used to set up encryption between networked devices. During a TLS connection, the client (your application) and server (Bronto) exchange messages using the same set of algorithms (ciphers).

Earlier versions of TLS can leave your server vulnerable to attack and are no longer recommended by groups like the PCI Security Standards Council due to security concerns.

Bronto requires using TLS versions of 1.2 or higher with the following cipher suites:

- AES256-SHA256
- AES256-SHA

- AES128-SHA256
- AES128-SHA
- AES256-GCM-SHA384
- AES128-GCM-SHA256
- AES256-SHA256
- AES256-SHA
- AES128-SHA256
- AES128-SHA
- ECDHE-RSA-AES256-SHA384
- ECDHE-RSA-AES256-CBC-SHA
- ECDHE-RSA-AES128-SHA256
- ECDHE-RSA-AES128-CBC-SHA
- ECDHE-RSA-AES256-GCM-SHA384
- ECDHE-RSA-AES128-GCM-SHA256
- ECDHE-RSA-AES256-SHA384
- ECDHE-RSA-AES256-CBC-SHA
- ECDHE-RSA-AES128-SHA256
- ECDHE-RSA-AES128-CBC-SHA
- DHE-RSA-AES256-GCM-SHA384
- DHE-RSA-AES128-GCM-SHA256
- DHE-RSA-AES256-SHA256
- DHE-RSA-AES256-SHA
- DHE-RSA-AES128-SHA256
- DHE-RSA-AES128-SHA

Use Bronto's API To Manage Consent Records

You can add or update consent records with the `addContacts`, `updateContacts`, and/or `addOrUpdateContacts` calls. The criteria for whether a new consent log entry will be created varies depending on which API method you use.

addContacts

Creating a new contact with `addContacts` will always create a new consent log entry. In your `addContacts` call, you can set both the `consentIp` and `consentDate` for a new contact.

updateContacts

The `updateContacts` call will only generate a new consent log entry if you modify the contact's status or email address. Updates to contacts to perform actions like changing a last name will not generate a new consent log entry, even if you have specified a `consentIp`. In addition, you cannot update the `consentDate` using `updateContacts` and can only change the `consentIp`. The `consentDate` will be the timestamp of the change to your contact's status.

addOrUpdateContacts

The `addOrUpdateContacts` function will only generate a new consent log entry if you modify an existing contact's email address or add a new contact. If you want to change the status of a contact, you must use `updateContacts`. If you create a new contact, you can set both the `consentIp` and `consentDate`. When you update an existing contact, you can only change the `consentIp`. The `consentDate` for an existing contact is the contact's status change timestamp.

SOAP API

Bronto's API was built on the SOAP web service, so most of the interactions you have with Bronto's API will be SOAP-based.

Here you will find the information you need to work with contacts, deliveries, messages (including SMS), lists, and logins. Our SOAP API library is divided into an Object reference section and a Function reference section. Functions are further broken up by the type of action a function performs.



Note: If you are working with product, order service, or cart data you will need to reference the REST API library to find the information you need. Most of the code samples in our SOAP library are written in PHP.

All API calls require the existence of an active, authenticated session. Therefore, before you can use our API reference library to interact with Bronto's API you will need to enable API access using the platform and set up authentication that you can use to log in.

Permissions

All API calls require that the user has permission to log in to the requested account. The user must also be designated as having the "API Access" permission. When using the API, this permission supersedes any other permissions granted to the user. Thus, API users are able to view and manipulate all account data that is accessible from within a web call.

Location

The WSDL specification for the API may be found at <https://api.bronto.com/v4?wsdl>.

API Tokens

Before logging into an account, the account must have the API feature enabled, and must have at least one API token created. API tokens act as an authentication and control object, and are similar to users. API tokens can have any combination of read, write, and sending permissions. They can be activated, deactivated, or deleted at any time. API tokens can be created programmatically via the API, or in the application.

If you have a Professional or Core edition account, you can create and edit API tokens by going to **Home > Settings > Data Exchange** in the platform.

Multi-Brand edition clients, and clients with certain versions of the Agency edition can create and edit tokens able to access all your accounts or a specific account. To create and edit API tokens in a Multi-Brand account, go to **Home > Settings > Data Exchange** at the Multi-Brand level.

Login To The SOAP API

All API calls require the existence of an active, authenticated session.

The function `login()` is used to create such a session. You pass an API token to login as credentials to authenticate your access. Upon successful authentication, a `sessionId` is returned.

The `sessionId` is a unique id used to identify an API session. API sessions become inactive if no calls are made for 20 minutes. Multiple sessions can be created for a single API token; If you wish to create a threaded application we recommend you create multiple sessions rather than reuse a single session in several threads. Successive calls to `login()` create a new session with no affect on previously created sessions.

PHP Code Example

```
<?php
/*
This example script will connect to the API, authenticate a session,
add a new contact, add a new list, update the contact to be on the
list, and then read the contact to verify that they are on the list.
```

```

Be sure to replace the "ADD YOUR API TOKEN HERE" text with a working
API Token.
*/
ini_set("soap.wsdl_cache_enabled", "0");
date_default_timezone_set('America/New_York');

$wsdl = "https://api.bronto.com/v4?wsdl";
$url = "https://api.bronto.com/v4";

$client = new SoapClient($wsdl, array('trace' => 1, 'encoding' => 'UTF-8'));
$client->__setLocation($url);

// Login
$token = "ADD YOUR API TOKEN HERE";
$sessionId = $client->login(array("apiToken" => $token))->return;
$client->__setSoapHeaders(array(new SoapHeader("http://api.bronto.com/v4",
    'sessionHeader',
    array('sessionId' =>

    $sessionId))));

// Add our new contact
$contact = array(
    array("email" => "john.doe@example.com")
);
$result = $client->addContacts(array("contacts" => $contact))->return-
>results;
if ($result->isError) {
    echo "Unable to add new contact!\n";
    return;
}
echo "Added new contact.\n";

// Add our new list
$list = array(
    array("name" => "My first list", "label" => "Hello Bronto List")
);
$result = $client->addLists(array("lists" => $list))->return->results;
if ($result->isError) {
    echo "Unable to add new list!\n";
    return;
}
$listId = $result->id;
echo "Added new list.\n";

// Update the contact to put him on the list
$contact = array(
    array("email" => "john.doe@example.com", "listIds" => array($listId))
);
$result = $client->updateContacts(array("contacts" => $contact))->return-
>results;
if ($result->isError) {
    echo "Unable to update contact!\n";
    return;
}
echo "Added contact to list.\n";

// Verify the contact is on the list
$filter = array(
    "email" => array(array("operator" => "EqualTo", "value"
=> "john.doe@example.com"))

```

```
);

$result = $client->readContacts(array(
    "filter" => $filter,
    "includeLists" => true,
    "fields" => null,
    "pageNumber" => 1
))->return;
if ($result->listIds != $listId) {
    echo "Contact is not on the list!\n";
    return;
}

echo "Contact is on the list.\n";
?>
```

Python Code Example

```
import sys
import logging
from suds.client import Client
from suds import WebFault

"""
This example script will login to the API, obtain
a session id, and then print the ID and email address
of all contacts whose email address contains the
string 'gmail'. BE SURE TO REPLACE ALL PLACEHOLDER TEXT

Tested with Python 2.6.1 and suds soap library version 0.4

See suds home page:
https://fedorahosted.org/suds/
"""

# Bronto API WSDL
BRONTO_WSDL = 'https://api.bronto.com/v4?wsdl'

# start up basic logging
logging.basicConfig()

# Replace the placeholder text with a valid
# API token
TOKEN = "ADD TOKEN HERE"

# login using the token to obtain a session ID
bApi = Client( BRONTO_WSDL )

try:
    session_id = bApi.service.login(TOKEN)
    # Just exit if something goes wrong
except WebFault, e:
    print '\nERROR MESSAGE:'
    print e
    sys.exit()

# Set up the soap headers using the
# session_id obtained from login()
session_header = bApi.factory.create("sessionHeader")
```

```

session_header.sessionId = session_id
bApi.set_options(soapheaders=session_header)

# Create the mailListFilter passed into
# readLists()

filter = bApi.factory.create('contactFilter')

stringValue = bApi.factory.create('stringValue')
stringValue.value = 'gmail'
filterOperator = bApi.factory.create('filterOperator')
stringValue.operator = filterOperator.Contains

filterType = bApi.factory.create('filterType')

filter.email = stringValue
filter.type = filterType.AND

# Try calling readContacts. Print email address and ID.
try:
    read_contact = bApi.service.readContacts(filter, includeLists = True,
        pageNumber = 1)
    for contact in read_contact:
        print 'Contact email: ' + contact.email
        print 'Contact ID: ' + contact.id
except WebFault, e:
    print '\nERROR MESSAGE: '
    print e
    sys.exit()

```

Hello Bronto

This section's goal is to help you write your first client to the Bronto API.

These simple example clients connect to the API, authenticate, add an example contact, add an example list, place the contact on the list, and then read back to the content for display to verify the operation.

Before you begin, remember that you will need an API token set up in your account with both reading and writing permissions. For more information on API tokens see [SOAP API](#) on page 5.

PHP Code Example

```

<?php

/*
This example script will connect to the API, authenticate a session,
add a new contact, add a new list, update the contact to be on the
list, and then read the contact to verify that they are on the list.
Be sure to replace the "ADD YOUR API TOKEN HERE" text with a working
API Token.
*/

ini_set("soap.wsdl_cache_enabled", "0");
date_default_timezone_set('America/New_York');

$wsdl = "https://api.bronto.com/v4?wsdl";
$url = "https://api.bronto.com/v4";

$client = new SoapClient($wsdl, array('trace' => 1, 'encoding' => 'UTF-8'));
$client->__setLocation($url);

// Login
$token = "ADD YOUR API TOKEN HERE";

```



```

$sessionId = $client->login(array("apiToken" => $token))->return;
$client->__setSoapHeaders(array(new SoapHeader("http://api.bronto.com/v4",
                                                'sessionHeader',
                                                array('sessionId' =>
$sessionId))));

// Add our new contact
$contact = array(
    array("email" => "john.doe@example.com")
);
$result = $client->addContacts(array("contacts" => $contact))->return-
>results;
if ($result->isError) {
    echo "Unable to add new contact!\n";
    return;
}
echo "Added new contact.\n";

// Add our new list
$list = array(
    array("name" => "My first list", "label" => "Hello Bronto List")
);
$result = $client->addLists(array("lists" => $list))->return->results;
if ($result->isError) {
    echo "Unable to add new list!\n";
    return;
}
$listId = $result->id;
echo "Added new list.\n";

// Update the contact to put him on the list
$contact = array(
    array("email" => "john.doe@example.com", "listIds" => array($listId))
);
$result = $client->updateContacts(array("contacts" => $contact))->return-
>results;
if ($result->isError) {
    echo "Unable to update contact!\n";
    return;
}
echo "Added contact to list.\n";

// Verify the contact is on the list
$filter = array(
    "email" => array(array("operator" => "EqualTo", "value"
=> "john.doe@example.com"))
);

$result = $client->readContacts(array(
    "filter" => $filter,
    "includeLists" => true,
    "fields" => null,
    "pageNumber" => 1
))->return;
if ($result->listIds != $listId) {
    echo "Contact is not on the list!\n";
    return;
}

echo "Contact is on the list.\n";

```

?>

Python Code Example

```
import sys
import logging
from suds.client import Client
from suds import WebFault

"""
This example script completes the following tasks:
1. Login to the API
2. Obtain a session ID
3. Add a contact
4. Add a list
5. Update the contact you just created and add them
   to the list you just created.
6. Read back the contact's data to ensure they were
   properly updated.

BE SURE TO REPLACE ALL PLACEHOLDER TEXT

Tested with Python 2.6.1 and suds soap library version 0.4

See suds home page:
https://fedorahosted.org/suds/
"""

# Bronto API WSDL
BRONTO_WSDL = 'https://api.bronto.com/v4?wsdl'

# Start up basic logging
logging.basicConfig()

# Replace the placeholder text with a valid
# API token
TOKEN = "ADD TOKEN HERE"

# Login using the token to obtain a session ID
bApi = Client( BRONTO_WSDL )

try:
    session_id = bApi.service.login(TOKEN)
    # Just exit if something goes wrong
except WebFault, e:
    print '\nERROR MESSAGE: '
    print e
    sys.exit()

# Set up the soap headers using the
# session_id obtained from login()
session_header = bApi.factory.create("sessionHeader")
session_header.sessionId = session_id
bApi.set_options(soapheaders=session_header)

# Adding a contact
# Be sure to replace the placeholder text with a
# real email address!
```

```

contact = bApi.factory.create('contactObject')
contact.email = 'SOME EMAIL ADDRESS'

try:
    add_contact = bApi.service.addContacts(contact)
    if add_contact.results[0].isError == True:
        print 'There was an error with your request:'
        print add_contact.results[0]
        sys.exit()
    else:
        print 'A contact has been added with the id: ' +
        add_contact.results[0].id
except WebFault, e:
    print '\nERROR MESSAGE:'
    print e
    sys.exit()

# Add a list
# Be sure to replace the placeholder text with a
# real list name and label!

list = bApi.factory.create('mailListObject')
list.name = 'example_list_name'
list.label = 'Example List'

try:
    add_list = bApi.service.addLists(list)
    if add_list.results[0].isError == True:
        print 'There was an error with your request:'
        print add_list.results[0]
        sys.exit()
    else:
        print 'A list has been created with an id of: ' + add_list.results[0].id
except WebFault, e:
    print '\nERROR MESSAGE:'
    print e
    sys.exit()

# Update the contact you just created contact to be
# on the list you just created list

update_contact = bApi.factory.create('contactObject')
update_contact.id = add_contact.results[0].id
update_contact.listIds = add_list.results[0].id

try:
    updated_contact = bApi.service.updateContacts(update_contact)
    if updated_contact.results[0].isError == True:
        print 'There was an error with your request:'
        print updated_contact.results[0]
        sys.exit()
    else:
        print 'The contact with an id of: ' + updated_contact.results[0].id + '
        has been updated.'
except WebFault, e:
    print '\nERROR MESSAGE:'
    print e
    sys.exit()

# Read back the contact data

filter = bApi.factory.create('contactFilter')

```

```

filterType = bApi.factory.create('filterType')

filter.id = add_contact.results[0].id
filter.type = filterType.AND

# Call readContacts. Return all the data for the contact
try:
    read_contact = bApi.service.readContacts(filter, includeLists = True,
    pageNumber = 1)
    for contact in read_contact:
        print contact
except WebFault, e:
    print '\nERROR MESSAGE:'
    print e
    sys.exit()

```

SOAP Add Functions

Add functions help you add new content to the Bronto platform. This can include things like adding contacts, delivery groups, lists, messages, etc.

If you want to update existing Bronto data you should use the update functions.

Add Accounts *Agency Only*

The addAccounts function allows you to add a new account and assign all the account settings.

Overview

Your API token must belong to an Agency, Groups, or Partner Edition master account in order to create new accounts. New accounts will be added with a status of active. If you wish to update the status of an account, use updateAccounts.

Syntax

```
writeResult = bApi.addAccounts(accountObject[] accounts);
```

PHP Code Example

```

<?php

/**
 * This example will add an Agency client account and
 * print out id of the account when it is created.
 */

$client = new SoapClient('https://api.bronto.com/v4?wsdl', array('trace' =>
1,
                        'features' => SOAP_SINGLE_ELEMENT_ARRAYS));

try {
    $token = "ADD YOUR TOKEN HERE";

    print "logging in\n";
    $sessionId = $client->login(array('apiToken' => $token))->return;

    $session_header = new SoapHeader("http://api.bronto.com/v4",
        'sessionHeader',
        array('sessionId' => $sessionId));
    $client->__setSoapHeaders(array($session_header));

```

```

//General Settings
$generalSettings = array(
    'agencyTemplateuploadPerm' => 1,
    'defaultTemplates' => 1,
    'enableInboxPreviews' => 0,
    'allowCustomizedBranding' => 1,
    'bounceLimit' => 7,
    'usageAlertEmail' => 'joe@example.com',
    'sendUsageAlerts' => 1,
);
$contactInfo = array(
    'organization' => 'ExampleCompany',
    'firstName' => 'Joe',
    'lastName' => 'Example',
    'email' => 'joe@example.com',
    'phone' => 5555555555,
    'address' => '312 Anywhere Street',
    'address2' => 'Suite 410',
    'city' => 'Durham',
    'state' => 'NC',
    'zip' => '27604',
    'country' => 'US',
    'notes' => 'This account was created via the API!',
);
$allocations = array(
    'startDate' => '2010-08-15T19:20:30-05:00',
    'periodFrequency' => 12,
    'emails' => 5000,
    'api' => 1,
    'bundle' => 'professional',
    'canExceedAllocation' => 1,
    'contacts' => 10000000,
    'hosting' => 100,
    'logins' => 10,
    'fields' => 100,
);
$account = array(
    'name' => 'ExampleAPIAccount4',
    'generalSettings' => $generalSettings,
    'contactInformation' => $contactInfo,
    'allocations' => $allocations,
);

$res = $client->addAccounts(array($account))->return;

if ($res->errors) {
    print "There was a problem adding the account:\n";
    print_r($res->results);
} else {
    print "Account has been created. Id: " . $res->results[0]->id . "\n";
}
} catch (Exception $e) {
    print "uncaught exception\n";
    print_r($e);
}
}

```

addApiTokens

The addApiTokens function allows you to add a new API token.

Syntax

```
writeResult = bApi.addApiTokens(apiTokenObject[] apiTokens);
```

Required and Optional API Token Object Attributes

Name	Type	Required	Description
name	string	Yes	The name assigned to API token. The name can be used to reference a specific API token when using the <code>apiToken</code> functions.
permissions	int	Yes	The permissions assigned to the API token. An API token can have read, write, and send permissions. Each permission is assigned an int value. To assign: <ul style="list-style-type: none"> • Read = 1 • Write = 2 • Read, Write = 3 • Send = 4 • Read, Send = 5 • Send, Write = 6 • Read, Write, Send = 7
active	boolean	Yes	Whether or not the API token is active. You can always go back and activate or deactivate API tokens at a later time.
accountId	boolean	Yes	The account, referenced by ID, that the API token is assigned to.

PHP Code Example

```
<?php

/**
 * This script will add an API token with read, write, and send
 * permissions. The API token will also be made active.
 *
 * @copyright Copyright (c) 2011 Bronto Software (http://www.bronto.com)
 */

$client = new SoapClient('https://api.bronto.com/v4?wsdl', array('trace' =>
    1,
    'features' => SOAP_SINGLE_ELEMENT_ARRAYS));

try {
    // Add your API token
```

```

$token = "API TOKEN HERE";

print "logging in\n";
$sessionId = $client->login(array('apiToken' => $token))->return;

$session_header = new SoapHeader("http://api.bronto.com/v4",
    'sessionHeader',
    array('sessionId' => $sessionId)
);
$client->__setSoapHeaders(array($session_header));

// NOTE the permissions parameter is an int. We use
// 7 to assign read, write, and send permissions.
// Replace the accountId place holder with a
// valid account ID.

$apiTokenObject = array('name' => 'Example API Token',
    'permissions' => '7',
    'active' => true,
    'accountId' => 'SOME ACCOUNT ID'
);
$write_result = $client->addApiTokens(array($apiTokenObject))->return;

// Note we are accessing the results and errors arrays.
// Both of these arrays are returned as part of
// writeResult object.

if ($write_result->errors) {
    print "There was a problem adding the account:\n";
    print_r($write_result->results);
} else {
    print "API Token has been created. Id: " . $write_result->results[0]-
>id . "\n";
}

} catch (Exception $e) {
    print "uncaught exception\n";
    print_r($e);
}
?>

```

addContactEvent

The addContactEvent function allows you to add contacts to any workflows containing a Received API Event trigger which also contains the specified keyword.

Overview

You can only add contacts to workflows which contain the Received API Event trigger and a keyword specified for that node.

Syntax

```
writeResult = bApi.addContactEvent(keyword, contactObject[] contacts);
```

Required and Optional Attributes

Name	Type	Required	Description
keyword	string	Yes if no workflow name is specified	The keyword assigned to the Received API Event trigger in a workflow or workflows.

Required and Optional Contact Object Attributes

Name	Type	Required	Description
id	string	Yes if no email address is specified	The unique id for the contact. The id can be used to reference a specific contact when using the contact functions. You can obtain the id for a contact by calling readContacts , or by looking at the footer when viewing the overview page for an individual contact in the application.
email	string	Yes if no contact id is specified	The email address assigned to the contact. The email address can be used to reference a specific contact when using the contact functions.

PHP Code Example

```

<?php
/**
 * This example will obtain the id for a contact and then
 * add that contact to a workflow by calling
 * addContactEvent. In order for a contact to
 * be added to a workflow, the workflow must contain
 * the Received API Event trigger node. The Received
 * API Event trigger node contains a keyword. This
 * keyword is used to reference that workflow using
 * API call. Note that more than one workflow can have
 * the same keyword. In this case, the contact will be
 * to each workflow containing the specified keyword.
 */

$client = new SoapClient('https://api.bronto.com/v4?wsdl', array('trace'
-----> 1,
                                'features' => SOAP_SINGLE_ELEMENT_ARRAYS));

try {
    // Add in a valid API token
    $token = "ADD API TOKEN HERE";

    print "logging in\n";
    $sessionId = $client->login(array('apiToken' => $token))->return;

```



```

$session_header = new SoapHeader("http://api.bronto.com/v4",
    'sessionHeader',
    array('sessionId' => $sessionId));
$client->__setSoapHeaders(array($session_header));

// Obtain the contact id for the contact being added to
// the workflow. You can always add more than 1 contact
// using addContactEvent if need be.

// Set up a filter to read contacts
$filter = array('type' => 'AND',
    'email' => array('operator' => 'EqualTo',
        'value' => 'SOME EXAMPLE EMAIL ADDRESS')
    );

print "reading contacts with equalto filter\n";
$contacts = $client->readContacts(array('pageNumber' => 1,
    'includeLists' => false,
    'filter' => $filter,
    )
)->return;

if (!$contacts) {
    print "There was an error reading your contacts. Please review your
    request and try again.\n";
    exit();
}

// Specify the keyword assigned to the Received API
// Event trigger node(s) added to the workflow(s) you want
// to add contacts to.
$keyword = 'SOME EXAMPLE KEYWORD';

print "Adding contacts to the workflow\n";

$write_result = $client->addContactEvent(array('keyword' => $keyword,
    'contacts' => $contacts)
)->return;

if ($write_result->errors) {
    print "There was a problem adding the contacts to the workflow:\n";
    print_r($write_result->results . "\n");
} else {
    print "The contacts have been added to the workflow.";
    print_r($write_result->results);
}

} catch (Exception $e) {
    print "uncaught exception\n";
    print_r($e);
}
?>

```

Python Code Example

```

import sys
import logging
from suds.client import Client
from suds import WebFault

```

```

# This example script will login in to the API, add
# a contact to a workflow. In order for a contact to
# be added to a workflow, the workflow must contain
# the Received API Event trigger node. The Received
# API Event trigger node contains a keyword. This
# keyword is used to reference that workflow using
# API call. Note that more than one workflow can have
# the same keyword. In this case, the contact will be
# to each workflow containing the specified keyword.

# BE SURE TO REPLACE ALL PLACEHOLDER TEXT

# Tested with Python 2.6.1 and suds soap library version 0.4

# See suds home page:
# https://fedorahosted.org/suds/

# Bronto API WSDL
BRONTO_WSDL = 'https://api.bronto.com/v4?wsdl'

# Start up basic logging
logging.basicConfig()

# Replace the placeholder text with a valid
# API token
TOKEN = "ADD A VALID TOKEN HERE"

# Login using the token to obtain a session ID
bApi = Client( BRONTO_WSDL )

try:
    session_id = bApi.service.login(TOKEN)
    # Just exit if something goes wrong
except WebFault, e:
    print '\nERROR MESSAGE: '
    print e
    sys.exit()

# Set up the soap headers using the
# session_id obtained from login()
session_header = bApi.factory.create("sessionHeader")
session_header.sessionId = session_id
bApi.set_options(soapheaders=session_header)

# Get the ID of the contact you wish to
# add to the workflow by calling readContacts.
# Be sure to replace the placeholder text below.
filter = bApi.factory.create("contactFilter")

# This example will pass in a single contact to
# the addContactEvent function, but you could
# add multiple contacts to the workflow if need be.
contact_email = bApi.factory.create('stringValue')
filter_operator = bApi.factory.create('filterOperator')
contact_email.operator = filter_operator.EqualTo
contact_email.value = 'SOME EXAMPLE EMAIL ADDRESS'

filterType = bApi.factory.create('filterType')

```

```

filter.email = contact_email
filter.type = filterType.AND

try:
    read_contact = bApi.service.readContacts(filter, pageNumber = 1)
    print 'The id for the contact: ' + read_contact[0].id
except WebFault, e:
    print '\nERROR MESSAGE: '
    print e
    sys.exit()

# Create a contact objec with the id returned
# from the previous readContacts call
contactObject = bApi.factory.create('contactObject')
contactObject.id = read_contact[0].id

# Specify the keyword assigned to the Received API
# Event trigger node(s) added to the workflow(s) you want
# to add contacts to.
keyword='SOME KEYWORD'

try:
    addContactEvent = bApi.service.addContactEvent(keyword, contactObject)
    if addContactEvent.results[0].isError == True:
        print 'There was an error with your request: '
        print addContactEvent.results[0]
        sys.exit()
    else:
        print 'The contact has been successfully added to the workflow.'
except WebFault, e:
    print '\nERROR MESSAGE: '
    print e
    sys.exit()

```

addContacts

The addContacts function allows you to add a new contact and data associated with that contact, such as field data and list membership.

Syntax

```
writeResult = bApi.addContacts(contactObject[] contacts);
```

Required and Optional Contact Object Attributes

Name	Type	Required	Description
email	string	No if mobileNumber is provided	The email address assigned to the contact. The email address can be used to reference a specific contact when using the contact functions.
mobileNumber	string	No if an email is provided	The mobile number for the contact. A valid country code must be included when adding a mobile number for a contact.

Name	Type	Required	Description
status	string	No; default is onboarding	The status of the contact. The status is automatically set to onboarding, unless you specifically set the status as unconfirmed or transactional.
msgPref	string	No; default is html	The message preference for the contact. A contact can have a message preference of text or html.
source	string	No; default is api	The source or where the contact came from. The source can be manual, import, api, webform, or sfocrereport (salesforce report).
customSource	string	No; default is empty	A source you define that states where the contact came from.
listIds	string, array. Use an array for multiple ids	No	The lists (referenced by ID) that the contact belongs to. You obtain listIds by calling the readLists function.
fields	contactField[]	No	An array of the fields and corresponding field data associated with the contact.
SMSKeywordIDs	string, array. Use an array for multiple ids	No	An array of the SMS keyword ids you want to subscribe the contact to.

PHP Code Example

```

<?php
/**
 * This script will add a contact, add that contact
 * to a list, and add field data associated with the
 * contact.
 *
 * @copyright Copyright (c) 2011 Bronto Software (<a href="http://
www.bronto.com" title="http://www.bronto.com">http://www.bronto.com</a>)
 */

$client = new SoapClient('https://api.bronto.com/v4?wsdl', array('trace' =>
1,
                                                                    'features'
=> SOAP_SINGLE_ELEMENT_ARRAYS));

try {
    // Add your API token

```

```

$token = "API TOKEN HERE";

print "logging in\n";
$sessionId = $client->login(array('apiToken' => $token))->return;

$session_header = new SoapHeader("http://api.bronto.com/v4",
                                'sessionHeader',
                                array('sessionId' => $sessionId));
$client->__setSoapHeaders(array($session_header));

// Replace SOME CONTENT with a string. We assume here
// the field is storing a string. The value you pass in
// should match the type set for the field.
// Replace SOME FIELD ID with a valid field ID. Field IDs
// can be obtained by calling readFields. Field IDs are also
// available in the footer when viewing an individual field in
// the UI.
$field1 = array('fieldId' => 'SOME FIELD ID',
               'content' => 'SOME CONTENT');
$field2 = array('fieldId' => 'SOME FIELD ID',
               'content' => 'SOME CONTENT');

// Add a contact, assign them to a specific list,
// and update some field data about the contact.
// NOTE: Replace mock email and listIds below.
// The status, msgPref, source, and customSource
// will use default values since we are not
// specifying them.
$contacts = array('email' => 'some_contact@example.com',
                  'listIds' => 'SOME LIST ID',
                  'fields' => array($field1, $field2)
                  );

print "Adding contact with the following attributes\n";
$write_result = $client->addContacts(array($contacts)
                                   )->return;

// Note we are accessing the results and errors arrays.
// Both of these arrays are returned as part of
// writeResult object.
if ($write_result->errors) {
    print "There was a problem adding the contact:\n";
    print_r($write_result->results);
} else {
    print "The contact has been created. Id: " . $write_result->
    >results[0]->id . "\n";
}

} catch (Exception $e) {
    print "uncaught exception\n";
    print_r($e);
}
?>

```

Python Code Example

```

import sys
import logging
from suds.client import Client
from suds import WebFault

```

```

# This example script will login in to the API, add
# a contact, add them to a list, and add some field
# data.

# BE SURE TO REPLACE ALL PLACEHOLDER TEXT

# Tested with Python 2.6.1 and suds soap library version 0.4

# See suds home page:
# https://fedorahosted.org/suds/

# Bronto API WSDL
BRONTO_WSDL = 'https://api.bronto.com/v4?wsdl'

# Start up basic logging
logging.basicConfig()

# Replace the placeholder text with a valid
# API token
TOKEN = "ADD API TOKEN HERE"

# Login using the token to obtain a session ID
bApi = Client( BRONTO_WSDL )

try:
    session_id = bApi.service.login(TOKEN)
    # Just exit if something goes wrong
except WebFault, e:
    print '\nERROR MESSAGE: '
    print e
    sys.exit()

# Set up the soap headers using the
# session_id obtained from login()
session_header = bApi.factory.create("sessionHeader")
session_header.sessionId = session_id
bApi.set_options(soapheaders=session_header)

# Set up the contactField objects
# Replace SOME CONTENT with a string. We assume here
# the field is storing a string. The value you pass in
# should match the type set for the field.
# Replace SOME FIELD ID with a valid field ID. Field IDs
# can be obtained by calling readFields. Field IDs are also
# available in the footer when viewing an individual field in
# the UI.
field1 = bApi.factory.create('contactField')
field1.fieldId = "SOME FIELD ID"
field1.content = "SOME CONTENT"

field2 = bApi.factory.create('contactField')
field2.fieldId = "SOME FIELD ID"
field2.content = "SOME CONTENT"

# Adding a contact, assigning them to a list,
# and adding some field data.
# Be sure to replace the placeholder text with a
# real email address and list ID!

contact = bApi.factory.create('contactObject')

```

```

contact.email = 'SOME EMAIL ADDRESS'
contact.listIds = 'SOME LIST ID'
contact.fields = [field1, field2]

try:
    add_contact = bApi.service.addContacts(contact)
    if add_contact.results[0].isError == True:
        print 'There was an error with your request:'
        print add_contact.results[0]
        sys.exit()
    else:
        print 'A contact has been added with the id: ' +
add_contact.results[0].id
except WebFault, e:
    print '\nERROR MESSAGE:'
    print e
    sys.exit()

```

addContactsToWorkflow

The `addContactsToWorkflow` function allows you to add contacts to a workflow that contains the Received API Event trigger.

Overview

You can only add contacts to workflows which contain the Received API Event trigger.

Syntax

```
writeResult = bApi.addContacts(contactObject[] contacts);
```

Required and Optional Workflow Object Attributes

Name	Type	Required	Description
id	string	Yes if no workflow name is specified	The unique id assigned to the workflow. You can obtain the id for a workflow by calling readWorkflow . The Received API Event trigger needs a workflow ID in order to trigger (start) the workflow.

Required and Optional Contact Object Attributes

Name	Type	Required	Description
id	string	Yes if no email address is specified	The unique id for the contact. The id can be used to reference a specific contact when using the contact functions. You can obtain the id for a contact by calling readContacts , or by looking at the footer when viewing the overview page for an individual contact in the application.
email	string	Yes if no contact id is specified	The email address assigned to the contact. The email address can be used to reference a specific contact when using the contact functions.

addContentTags

The `addContentTags` function allows you to add content tags to use in your account.

Overview

Content tags allow you to create reusable blocks of content that you can use in the body, header, and footer of your email messages. The block is referenced via a custom defined content tag you create. When the message is sent, the content tag is replaced with the appropriate content.

Syntax

```
writeResult = bApi.addContentTags(contentTagObject[] contentTags);
```


Required and Optional Content Tag Object Attributes

Name	Type	Required	Description
name	string	Yes	<p>The name you assigned to the content tag. The name you specify will be used to reference this block of content via the Content Tag. For example, if you name the Content Tag mycontenttag, you would reference this Content Tag in your message by adding <code>%%@mycontenttag%%</code> to your message. Note:</p> <ul style="list-style-type: none"> The content tag name must be 100 characters or less. The name cannot be blank or null
value	string	Yes	<p>The content that will be displayed when the message is sent. Note:</p> <ul style="list-style-type: none"> The value cannot contain other content tags, field tags, API message tags, or dynamic code. HTML can be used in the value, however, the HTML will appear unformatted if the content tag is used in a plain text message.

PHP Code Example

```

<?php

/**
 * This script will add a content tag.
 * Be sure to replace all the generic placeholder text!
 *
 * @copyright Copyright (c) 2013 Bronto Software (http://www.bronto.com)
 */

$client = new SoapClient('https://api.bronto.com/v4?wsdl', array('trace' ===
> 1,
                                'features'
=> SOAP_SINGLE_ELEMENT_ARRAYS));

try {
    $token = "ADD API TOKEN HERE";

```

```

print "logging in\n";
$sessionId = $client->login(array('apiToken' => $token))->return;

$session_header = new SoapHeader("http://api.bronto.com/v4",
                                'sessionHeader',
                                array('sessionId' => $sessionId)
                                );
$client->__setSoapHeaders(array($session_header));

$contentTagObject = array('name' => 'EXAMPLE NAME',
                          'value' => 'EXAMPLE CONTENT'
                          );
$write_result = $client->addContentTags(array($contentTagObject))->return;

    if ($write_result->errors) {
        print "There was a problem adding the content tag:\n";
        print_r($write_result->results);
    } else {
        print "The content tag has been created.  Id: " .
$write_result->results[0]->id . "\n";
    }

} catch (Exception $e) {
    print "uncaught exception\n";
    print_r($e);
}

?>

```

Python Code Example

```

import sys
import logging
from suds.client import Client
from suds import WebFault

# This example script will login in to the API and add a
# content tag

# BE SURE TO REPLACE ALL PLACEHOLDER TEXT

# Tested with Python 2.7.1 and suds soap library version 0.4

# See suds home page:
# https://fedorahosted.org/suds/

# Bronto API WSDL
BRONTO_WSDL = 'https://api.bronto.com/v4?wsdl'

# Start up basic logging
logging.basicConfig()

# Replace the placeholder text with a valid
# API token
TOKEN = "API TOKEN HERE"

# Login using the token to obtain a session ID

```

```

bApi = Client( BRONTO_WSDL )

try:
    session_id = bApi.service.login(TOKEN)
    # Just exit if something goes wrong
except WebFault, e:
    print '\nERROR MESSAGE: '
    print e
    sys.exit()

# Set up the soap headers using the
# session_id obtained from login()
session_header = bApi.factory.create("sessionHeader")
session_header.sessionId = session_id
bApi.set_options(soapheaders=session_header)

# Adding a content tag
# Be sure to replace the placeholder text with a
# real email address!

contentTag = bApi.factory.create('contentTagObject')
contentTag.name = 'examplecontenttag'
contentTag.value = "Here is some example text"

try:
    addcontentTag = bApi.service.addContentTags(contentTag)
    if addcontentTag.results[0].isError == True:
        print 'There was an error with your request: '
        print addcontentTag.results[0]
        sys.exit()
    else:
        print 'A content tag has been added with the id: ' +
        addcontentTag.results[0].id
except WebFault, e:
    print '\nERROR MESSAGE: '
    print e
    sys.exit()

```


addConversion


The addConversion function allows you to add a conversion and any data associated with that conversion.


Syntax

```
writeResult = bApi.addConversion(conversionObject[] conversions);
```

Required and Optional Conversion Object Attributes

Name	Type	Required	Description
contactId	string	No	<p>The unique id assigned to the contact you want to add a conversion for.</p> <p> Note: The tid will take priority over other ids (deliveryId, contactId, email) passed in.</p>

Name	Type	Required	Description
email	string	No	The email address of the conversion contact.
orderId	string	No	The order identifier. This should be unique per order and will be used to prevent duplicate orders.
item	string	No	The SKU of the line item. Certain item codes are reserved for special use subtotal, taxes, shipping, and total.
description	string	No	The description of the line item.
quantity	int	Yes	The unit count of the line item.
amount	float	Yes	The grand total of the line items.
orderTotal	int	No	The total number of orders made.
createdDate	datetime	No	The date and time of the conversion. If no date/time is provided, then the system will timestamp the record. You can (and should) specify a timezone offset if you do not want the system to assume you are providing a time in UTC (Universal Coordinated Time / Greenwich Mean Time). For the Eastern Time Zone on Daylight Savings Time, this would be YYYY-MM-DDTHH:MM:SS-04:00.
deliveryId	string	No	<p>The unique ID assigned to a delivery you want to associate a conversion with.</p> <p> Note: The tid will take priority over other ids (deliveryId, contactId, email) passed in.</p>

Name	Type	Required	Description
messageId	string	No	The unique ID assigned to a message you want to associate a conversion with.
automatorId	string	No	The unique ID assigned to an automator you want to associate a conversion with.
listId	string	No	The unique ID assigned to a list you want to associate a conversion with.
segmentId	string	No	The unique ID assigned to a segment you want to associate a conversion with.
tid	string	No	<p>A unique id which associates a conversion with a specific contact and delivery. When conversion tracking is enabled in the application, you can pull the <code>tid</code> from the bronto tracking cookie and use it in the <code>addConversion</code> call to record a conversion that a specific contact makes from a specific delivery.</p> <p> Note: The <code>tid</code> will take priority over other ids (<code>deliveryId</code>, <code>contactId</code>, <code>email</code>) passed in.</p> <p>The <code>tid</code> is not returned when calling readConversion. In order to obtain the <code>tid</code>, you must pull it from the bronto tracking cookie.</p>

addDeliveries

The `addDeliveries` function allows you to add a new delivery/deliveries along with data associated with the delivery.

Overview

The `addDeliveries` function will return a delivery ID. Please wait a few minutes before attempting to read delivery data (`readDeliveries`) using the returned delivery ID.




Note: When you insert dynamic links in messages via the API, Bronto will only track up to 30 dynamic links for single send messages and up to 10 for bulk messages.

Syntax


```
writeResult = bApi.addDeliveries(deliveryObject[] deliveries);
```

Required and Optional Delivery Attributes

Name	Type	Required	Description
start	dateTime	Required for bulk sends only; optional for single sends	The date the delivery was scheduled to be sent. You can (and should) specify a timezone offset if you do not want the system to assume you are providing a time in UTC (Universal Coordinated Time / Greenwich Mean Time). For the Eastern Time Zone on Daylight Savings Time, this would be YYYY-MM-DDTHH:MM:SS-04:00. For single contact sends, start is not required.
messageId	string	Yes	The id of the message associated with the delivery. You must reference the message you want to use in the delivery by ID. We do not support passing in a messageContentObject when using addDeliveries.

Name	Type	Required	Description
type	string	Yes	<p>The type of delivery. Valid values are:</p> <ul style="list-style-type: none"> • triggered • test • transactional. <p>triggered is the default. Only use transactional when adding a delivery which uses a message that has been approved for transactional sending.</p> <p> Note: If you attempt to send a triggered or testdelivery to a contact with a status of transactional, the delivery will be set to skipped and the delivery will not be sent.</p>
fromEmail	string	Yes	The email address used in the From Address for this delivery.
fromName	string	Yes	The name used as the From Name for the delivery
replyEmail	string	No	The email address used as the Reply-To Address for the delivery. If no replyEmail is provided, the fromEmail will be used.

Name	Type	Required	Description
authentication	boolean	No	Enables sender authentication for the delivery if set to <code>true</code> . Sender authentication will sign your message with DomainKeys/DKIM (an authentication method that can optimize your message delivery to Hotmail, MSN, and Yahoo! email addresses). If you are associating this delivery with an automated message rule, these parameters will only be accepted if you clicked the Allow API to select sending options checkbox via the application UI on step 2 of creating an API triggered automated message rule.
replyTracking	boolean	No	Enables reply tracking for the delivery if set to <code>true</code> . Enabling Reply Tracking will store a copy of all replies to your messages on the Replies page. You may find this option convenient if you need someone other than the email address in the From line to read replies, or simply want the application to store replies. If you are associating this delivery with an automated message rule, these parameters will only be accepted if you clicked the Allow API to select sending options checkbox via the platform when creating an API triggered automated message rule.
messageRuleId	string	No	The ID of an automated message rule to associate with this delivery. Used to include this delivery in the reporting for the automator you specify.

Name	Type	Required	Description
optin	boolean	No; default is false	Whether or not this delivery is an opt-in confirmation email. If set to true, contacts who have not yet confirmed their opt-in status with the account will still receive the message.
throttle	long	No	<p>Allows you to specify a throttle rate for the delivery. Throttle rate must be in range [0, 720] (minutes). For example you could specify 60 for the throttle range.</p> <p> Note: Throttling slows your email delivery speed, and thus spreads your email deliveries out over time. Many major ISPs track a sender's reputation based on the number of unsolicited email complaints that they generate over a certain period of time. The worse your sender reputation is with ISPs, the more likely you are to have a low sender rating in the application. Spreading your email delivery over time can help mitigate the impact of this issue, help ensure optimal deliverability, and mitigate traffic spikes on your website.</p>

Name	Type	Required	Description
fatigueOverride	boolean	No	If set to <code>true</code> , the delivery can be sent even if it exceeds frequency cap settings for a contact.
reemail	reemailObject[]	No	A <code>reemail</code> object. Remails allow you to send another email to contacts based on actions they did not take. The goal is to persuade them to continue along the conversion process.
recipients	deliveryRecipientObject[]	Yes	An array of the recipients who were or are scheduled to receive the delivery. Recipients can include lists, segments, or individual contacts.
fields	messageFieldObject[]	No; default is none	An array of the API fields and data to substitute into the message being sent by this delivery.
products	deliveryProductObject[]	No; default is none	Specifies Product IDs to substitute for placeholders in product tags upon message send. Limit: 100 products
cartId	string	No	The ID of a shopping cart you want to associate with this delivery.
orderId	string	No	The Bronto-provided ID for an order you want to associate with this delivery.

PHP Code Example For Single Delivery

```

/**
 * This example will create a delivery to send a message to a single
 * contact.
 * You must edit the code to refer to the message name you wish to
 * send,
 * and change the email address of the contact to be a valid contact
 * in
 * your account.
 */

$client = new SoapClient('https://api.bronto.com/v4?
wsdl', array('trace' => 1,
            'features' => SOAP_SINGLE_ELEMENT_ARRAYS));
setlocale(LC_ALL, 'en_US');

try {
    $token = "YOUR_TOKEN_HERE";

```

```

    print "logging in\n";
    $sessionId = $client->login(array('apiToken' => $token))-
    >return;

    $session_header = new SoapHeader("http://api.bronto.com/v4",
        'sessionHeader',
        array('sessionId' => $sessionId));
    $client->__setSoapHeaders(array($session_header));

    // get the id of the message you wish to send. It would be more
    // efficient to hardcode the ID here, and you may choose to do that
    // based on your own usage scenario(s).
    $messageFilter = array('name' => array('operator'
=> 'EqualTo',
        'value' => 'NAME OF YOUR MESSAGE HERE')
    );
    $message = $client->readMessages(array('pageNumber' => 1,
        'includeContent' => false,
        'filter' => $messageFilter)
    )->return[0];
    if (!$message) {
        print "There was an error retrieving your message.\n";
        exit;
    }

    // get the id of the contact you will be sending to. It would be
    // more efficient to hardcode the ID here, and you may choose to do
    that
    // based on your own usage scenario(s).

    $contactFilter = array('email' => array('operator'
=> 'EqualTo',
        'value' => 'RECIPIENT_EMAIL@EXAMPLE.COM'
    ),);
    $contact = $client->readContacts(array('pageNumber' => 1,
        'includeLists' => false,
        'filter' => $contactFilter)
    )->return[0];
    if (!$contact) {
        print "There was an error retrieving your contact.\n";
        exit;
    }

    $deliveryRecipientObject = array('type' => 'contact',
        'id' => $contact->id);

    $delivery = array('messageId' => $message->id,
        'fromName' => 'Bronto API Robot',
        'fromEmail' => 'bronto_api_test@example.com',
        'recipients' => array($deliveryRecipientObject),
    );
    $res = $client->addDeliveries(array($delivery))->return;
    if ($res->errors) {
        print "There was a problem scheduling your delivery:\n";
        print $res->results[$res->errors[0]]-
    >errorString . "\n";
    } else {
        print "Delivery has been scheduled. Id: " . $res-
    >results[0]->id . "\n";
    }
} catch (Exception $e) {

```

```

        print "uncaught exception\n";
        print_r($e);
    }

```

PHP Code Example For Bulk Delivery

```

/**
 * This example will create a delivery to send a message to a list
 * using readLists
 * to retrieve the list id. You must edit the code to refer to the
 * message name you
 * wish to send, and change the list name to be a valid list in your
 * account.
 */

$client = new SoapClient('https://api.bronto.com/v4?
wsdl', array('trace' => 1,
            'features' => SOAP_SINGLE_ELEMENT_ARRAYS));
setlocale(LC_ALL, 'en_US');

try {
    $token = "ADD YOUR API TOKEN HERE";

    print "logging in\n";
    $sessionId = $client->login(array('apiToken' => $token))-
    &return;

    $session_header = new SoapHeader("http://api.bronto.com/v4",
        'sessionHeader',
        array('sessionId' => $sessionId));
    $client->__setSoapHeaders(array($session_header));

    // Get the id of the message you wish to send. It would be more
    // efficient to hardcode the ID here, and you may choose to do that
    // based on your own usage scenario(s).
    $messageFilter = array('name' => array('operator'
=> 'EqualTo',
            'value' => 'NAME OF YOUR MESSAGE HERE')
    );
    $message = $client->readMessages(array('pageNumber' => 1,
        'includeContent' => false,
        'filter' => $messageFilter)
    )->return[0];
    if (!$message) {
        print "There was an error retrieving your message.\n";
        exit;
    }

    // get the id of the list you will be sending to. It would be
    // more efficient to hardcode the ID here, and you may choose to do
    that
    // based on your own usage scenario(s). You can use segmentFilter
    // instead here if you are sending to a segment.
    $listFilter = array('name' => array('operator' => 'EqualTo',
        'value' => 'NAME OF YOUR LIST HERE'
    ));

    $list = $client->readLists(array('pageNumber' => 1,
        'includeLists' => false,
        'filter' => $listFilter)
    )->return[0];

```

```

    if (!$list) {
        print "There was an error retrieving your list.\n";
        exit;
    }

    // make delivery start timestamp
    $now = date('c');

    $deliveryRecipientObject = array('type' => 'list',
                                     'id' => $list->id);

    $delivery = array('start' => $now,
                     'messageId' => $message->id,
                     'fromName' => 'Bronto API Robot',
                     'fromEmail' => 'bronto_api_test@example.com',
                     'recipients' => array($deliveryRecipientObject),
                     );
    $res = $client->addDeliveries(array($delivery))->return;

    if ($res->results[0]->isError) {
        print "There was a problem scheduling your delivery:\n";
        print $res->results[0]->errorString . "\n";
    } else {
        print "Delivery has been scheduled. Id: " . $res->results[0]->id . "\n";
    }
} catch (Exception $e) {
    print "uncaught exception\n";
    print_r($e);
}

```

Python Code Example

```

import sys
import logging
from suds.client import Client
from suds import WebFault
from datetime import datetime

# This example script will login in to the API, obtain
# a message ID and contact ID, and use those IDs to
# send the message to the contact.

# BE SURE TO REPLACE ALL PLACEHOLDER TEXT

# Tested with Python 2.6.1 and suds soap library version 0.4

# See suds home page:
# https://fedorahosted.org/suds/

# Bronto API WSDL
BRONTO_WSDL = 'https://api.bronto.com/v4?wsdl'

# Start up basic logging
logging.basicConfig()

# Replace the placeholder text with a valid
# API token

```

```

TOKEN = "API TOKEN HERE"

# Login using the token to obtain a session ID
bApi = Client( BRONTO_WSDL )

try:
    session_id = bApi.service.login(TOKEN)
# Just exit if something goes wrong
except WebFault, e:
    print '\nERROR MESSAGE: '
    print e
    sys.exit()

# Set up the soap headers using the
# session_id obtained from login()
session_header = bApi.factory.create("sessionHeader")
session_header.sessionId = session_id
bApi.set_options(soapheaders=session_header)

# Get the ID of the message you wish to
# send by calling readMessages.
# Be sure to replace the placeholder text below

messageFilter = bApi.factory.create('messageFilter')

message_name = bApi.factory.create('stringValue')
filter_operator = bApi.factory.create('filterOperator')
message_name.operator = filter_operator.EqualTo
message_name.value = 'SOME MESSAGE NAME'
messageFilter.name = message_name

filterType = bApi.factory.create('filterType')
messageFilter.type = filterType.AND

try:
    read_message = bApi.service.readMessages(messageFilter, includeContent =
False, pageNumber = 1)
    print 'The id for the message: ' + read_message[0].id
except WebFault, e:
    print '\nERROR MESSAGE: '
    print e
    sys.exit()

# Get the ID of the contact you wish to send
# to by calling readContacts,
# Be sure to replace the placeholder text below
filter = bApi.factory.create('contactFilter')

stringValue = bApi.factory.create('stringValue')
stringValue.value = 'SOME EMAIL ADDRESS'
filterOperator = bApi.factory.create('filterOperator')
stringValue.operator = filterOperator.EqualTo

filterType = bApi.factory.create('filterType')

filter.email = stringValue
filter.type = filterType.AND

try:
    read_contact = bApi.service.readContacts(filter, includeLists = True,
pageNumber = 1)
    print 'Contact ID: ' + read_contact[0].id

```

```

except WebFault, e:
    print '\nERROR MESSAGE:'
    print e
    sys.exit()

# Get the current date and time so we can send the
# email message now
sendtime = datetime.now()

# Call addDeliveries using the send time, message ID, and
# contact ID specified above.
# Be sure to replace the placeholder text below

add_delivery = bApi.factory.create('deliveryObject')
add_delivery.start = sendtime
add_delivery.messageId = read_message[0].id
add_delivery.fromName = 'SOME FROM NAME'
add_delivery.fromEmail = 'SOME FROM EMAIL ADDRESS'

deliveryRecipientObject = bApi.factory.create('deliveryRecipientObject')
deliveryRecipientObject.type = 'contact'
deliveryRecipientObject.id = read_contact[0].id
add_delivery.recipients = deliveryRecipientObject

try:
    delivery = bApi.service.addDeliveries(add_delivery)
    if delivery.results[0].isError == True:
        print 'There was an error with your request:'
        print delivery.results[0]
        sys.exit()
    else:
        print 'The delivery has been sent. The delivery ID is: ' +
        delivery.results[0].id
except WebFault, e:
    print '\nERROR MESSAGE:'
    print e
    sys.exit()

```

Python Code Example With Loop Tags

```

import sys
import logging
from suds.client import Client
from suds import WebFault
from datetime import datetime

# This example script will login in to the API, obtain
# a message ID and contact ID, and use those IDs to
# send the message to the contact. This script assumes
# the email message being sent contains {loop}/{loop}
# tags and the following the API message tags contained
# within the loop tags:
# %%#productname_#%
# %%#productprice_#%

# BE SURE TO REPLACE ALL PLACEHOLDER TEXT

# Tested with Python 2.6.1 and suds soap library version 0.4

# See suds home page:

```

```

# https://fedorahosted.org/suds/

# Bronto API WSDL
BRONTO_WSDL = 'https://api.bronto.com/v4?wsdl'

# Start up basic logging
logging.basicConfig()

# Replace the placeholder text with a valid
# API token
TOKEN = "ADD API TOKEN HERE"

# Login using the token to obtain a session ID
bApi = Client( BRONTO_WSDL )

try:
    session_id = bApi.service.login(TOKEN)
    # Just exit if something goes wrong
except WebFault, e:
    print '\nERROR MESSAGE: '
    print e
    sys.exit()

# Set up the soap headers using the
# session_id obtained from login()
session_header = bApi.factory.create("sessionHeader")
session_header.sessionId = session_id
bApi.set_options(soapheaders=session_header)

# Get the ID of the message you wish to
# send by calling readMessages.
# Make sure the message contains {loop}/{/loop}
# tags and the following the API message tags contained
# within the loop tags:
#     %%#productname_#%
#     %%#productprice_#%
# Be sure to replace the placeholder text below

messageFilter = bApi.factory.create('messageFilter')

message_name = bApi.factory.create('stringValue')
filter_operator = bApi.factory.create('filterOperator')
message_name.operator = filter_operator.EqualTo
message_name.value = 'SOME MESSAGE NAME'
messageFilter.name = message_name

filterType = bApi.factory.create('filterType')
messageFilter.type = filterType.AND

try:
    read_message = bApi.service.readMessages(messageFilter, includeContent =
False, pageNumber = 1)
    print 'The id for the message: ' + read_message[0].id
except WebFault, e:
    print '\nERROR MESSAGE: '
    print e
    sys.exit()

# Get the ID of the contact you wish to send
# to by calling readContacts,

```



```

# Be sure to replace the placeholder text below
filter = bApi.factory.create('contactFilter')

stringValue = bApi.factory.create('stringValue')
stringValue.value = 'SOME EMAIL ADDRESS'
filterOperator = bApi.factory.create('filterOperator')
stringValue.operator = filterOperator.EqualTo

filterType = bApi.factory.create('filterType')

filter.email = stringValue
filter.type = filterType.AND

try:
    read_contact = bApi.service.readContacts(filter, includeLists = True,
    pageNumber = 1)
    print 'Contact ID: ' + read_contact[0].id
except WebFault, e:
    print '\nERROR MESSAGE: '
    print e
    sys.exit()

# Get the current date and time so we can send the
# email message now
sendtime = datetime.now()

# Call addDeliveries using the send time, message ID, and
# contact ID specified above.
# Be sure to replace the placeholder text below

add_delivery = bApi.factory.create('deliveryObject')
add_delivery.start = sendtime
add_delivery.messageId = read_message[0].id
add_delivery.fromName = 'SOME FROM NAME'
add_delivery.fromEmail = 'SOME FROM EXAMPLE'

deliveryRecipientObject = bApi.factory.create('deliveryRecipientObject')
deliveryRecipientObject.type = 'contact'
deliveryRecipientObject.id = read_contact[0].id
add_delivery.recipients = deliveryRecipientObject

fieldObject = bApi.factory.create('messageFieldObject')

add_delivery.fields = fieldObject

# Create a fieldObject list of each
# fieldName and fieldPrice
fieldObject = []

# For each fieldName and fieldPrice, create
# a dictionary containing a name, type, and content, and
# then append that dictionary to the fieldObject list.

fieldName0 = {}
fieldName0['name'] = 'productname_0'
fieldName0['type'] = 'html'
fieldName0['content'] = 'A Cool Shirt'
fieldObject.append(fieldName0)

fieldName1 = {}
fieldName1['name'] = 'productname_1'
fieldName1['type'] = 'html'
fieldName1['content'] = 'Some Nice Shoes'

```

```

fieldObject.append(fieldName1)

fieldName2 = {}
fieldName2['name'] = 'productname_2'
fieldName2['type'] = 'html'
fieldName2['content'] = 'A Trendy Hat'
fieldObject.append(fieldName2)

fieldPrice0 = {}
fieldPrice0['name'] = 'productprice_0'
fieldPrice0['type'] = 'html'
fieldPrice0['content'] = '20.99'
fieldObject.append(fieldPrice0)

fieldPrice1 = {}
fieldPrice1['name'] = 'productprice_1'
fieldPrice1['type'] = 'html'
fieldPrice1['content'] = '50.99'
fieldObject.append(fieldPrice1)

fieldPrice2 = {}
fieldPrice2['name'] = 'productprice_2'
fieldPrice2['type'] = 'html'
fieldPrice2['content'] = 'FREE'
fieldObject.append(fieldPrice2)

add_delivery.fields = fieldObject

try:
    delivery = bApi.service.addDeliveries(add_delivery)
    if delivery.results[0].isError == True:
        print 'There was an error with your request:'
        print delivery.results[0]
        sys.exit()
    else:
        print 'The delivery has been sent. The delivery ID is: ' +
        delivery.results[0].id
except WebFault, e:
    print '\nERROR MESSAGE:'
    print e
    sys.exit()

```

addDeliveryGroup

The addDeliveryGroup function allows you to add a delivery group.

Syntax

```
writeResult = bApi.addDeliveryGroup(deliveryGroupObject[] deliveryGroup);
```

Required and Optional Delivery Attributes

Name	Type	Required	Description
name	string	Yes	The name associated with the delivery group.

PHP Code Example

```
<?php
/**
 * This script will add a new delivery group.
 *
 * @copyright Copyright (c) 2018 Oracle + Bronto Software (http://
www.bronto.com)
 */

$client = new SoapClient('https://api.bronto.com/v4?wsdl', array('trace'
=> 1,
                                'features' =>
SOAP_SINGLE_ELEMENT_ARRAYS));
try {
    // Add your API token
    $token = "ADD YOUR API TOKEN";

    print "Logging in\n";
    $sessionId = $client->login(array('apiToken' => $token))->return;

    $session_header = new SoapHeader(
        "http://api.bronto.com/v4",
        'sessionHeader',
        array('sessionId' => $sessionId)
    );
    $client->__setSoapHeaders(array($session_header));

    // $deliveryGroupObject is an array containing the delivery group
    information.
    // The name is the name of your new delivery group.
    $deliveryGroupObject = array("name" => "NEW DELIVERY GROUP NAME");

    $write_result = $client-
>addDeliveryGroup(array($deliveryGroupObject))->return;
    if ($write_result->errors) {
        print "There was a problem adding the delivery group.\n";
        print_r($write_result->results);
    } else {
        print "The delivery group has been added.\n";
    }
} catch (Exception $e) {
    print "uncaught exception\n";
    print_r($e);
}
?>
```

addFields

The `addFields` function allows you to add new fields and field data.

Overview

You should not use fields to store particularly sensitive or private information about your contacts. Information such as credit card numbers, social security numbers, unencrypted passwords, and other similar data should be stored outside of the application in a system specifically designed for handling this type of data.

Any field data you use in a message must be smaller than 1 MB.

Syntax

```
writeResult = bApi.addFields(fieldObject[] fields);
```

Required and Optional Field Objects

Name	Type	Required	Description
name	string	Yes	The internal name of the field.
label	string	Yes	The external (public facing) name of the field.
type	string	Yes	<p>The type of field:</p> <ul style="list-style-type: none"> • <code>text</code> – Text box (max character limit: 65535) • <code>textarea</code> – A large multi-line text box (max character limit: 65535) • <code>password</code> – A text box that hides typed characters (max character limit: 65535) • <code>checkbox</code> – Checkboxes (max character limit: 1) • <code>radio</code> – Radio buttons • <code>select</code> – Pull-down menu • <code>integer</code> – Accepts any 10 digit whole number less than or equal to 2147483647. If you need to store a static number (i.e. a number that is not incrementing or decrementing) we suggest using either a predefined field, or a custom field with a type of Text. • <code>currency</code> – Accepts any positive or negative number with two decimal points (max character limit: 15) • <code>float</code> – Accepts any positive or negative number with a decimal point (max character limit: 53) • <code>date</code> – A value that matches a specific date. See SOAP Data Formats on page 249 for more information on the date format.
visibility	string	No; default is public	The visibility selected for the field {public, private}. Public fields are visible to you and can be made visible to your contacts. Private fields are visible only to you.
options	FieldOptionObject []	Not unless the type requires it	The possible options that can be set for a field if the field is a pull-down, check box, or radio button.

PHP Code Example

```
<?php
/**
 * This script will add add fields to an account.
 *
 * @copyright Copyright (c) 2018 Bronto Software (http://www.bronto.com)
 */
$client = new SoapClient('https://api.bronto.com/v4?wsdl', array(
    'trace' => 1,
```

```

'features' => SOAP_SINGLE_ELEMENT_ARRAYS
));
try {
    // Add your API token
    $token = "YOUR API KEY";
    print "logging in\n";
    $sessionId = $client->login(array(
        'apiToken' => $token
    ))->return;
    $session_header = new SoapHeader("http://api.bronto.com/
v4", 'sessionHeader', array(
        'sessionId' => $sessionId
    ));
    $client->__setSoapHeaders(array(
        $session_header
    ));
    // Adding a text field
    // Uncomment and comment out select $fieldObject to use
    // $fieldObject = array('name' => 'MY TEXT FIELD',
    //                      "label" => 'MY FIELD TEXT LABEL',
    //                      "type" => "text"
    //                      );
    $optionGreen = array(
        'value' => 'green',
        "label" => "Green",
        'isDefault' => false
    );
    $optionBlue = array(
        'value' => 'blue',
        "label" => "Blue",
        'isDefault' => false
    );
    $optionRed = array(
        'value' => 'red',
        "label" => "Red",
        'isDefault' => true
    );
    //Adding a select option. The values are passed in the options array
    $fieldObject = array(
        'name' => 'MY SELECT FIELD',
        "label" => 'MY FIELD SELECT LABEL',
        "type" => "select",
        "options" => array(
            $optionGreen,
            $optionBlue,
            $optionRed
        )
    );
    print "Adding the field\n";
    $write_result = $client->addFields(array(
        $fieldObject
    ))->return;
    if (isset($write_result->errors)) {
        print "There was a problem adding the field:\n";
        print_r($write_result->results);
    } else {
        print "The field has been added. Id: " . $write_result->results[0]-
>id . "\n";
    }
}
catch (Exception $e) {
    print "uncaught exception\n";
}

```

```

        print_r($e);
    }
    ?>

```

addHeaderFooters

The `addHeaderFooters` function allows you to add new reusable headers and footers that can be included at the top and bottom of messages.

Syntax

```
writeResult = bApi.addHeaderFooters(headerFooterObject[] headerFooters)
```

Required and Optional HeaderFooter Object Attributes

Name	Type	Required	Description
name	string	Yes	The name assigned to the header/footer.
html	string	Yes	The HTML version of the header.
text	string	Yes	The text version of the header
isHeader	boolean	Yes	Set to TRUE if the object is a header.

PHP Code Example

```

<?php
/**
 * This script will add a header
 *
 * @copyright Copyright (c) 2011 Bronto Software (http://www.bronto.com)
 */

$client = new SoapClient('https://api.bronto.com/v4?wsdl', array('trace' =>
    1,
                                'features' => SOAP_SINGLE_ELEMENT_ARRAYS));

try {
    // Add your API token
    $token = "ADD API TOKEN";

    print "logging in\n";
    $sessionId = $client->login(array('apiToken' => $token))->return;

    $session_header = new SoapHeader("http://api.bronto.com/v4",
        'sessionHeader',
        array('sessionId' => $sessionId));
    $client->__setSoapHeaders(array($session_header));

    // Add a header with some HTML and plain text content
    // isHeader should be set to false if you want to create
    // a footer.
    // Be sure to replace all place holder content with real
    // data.

    $headerFooterObject = array('name' => 'EXAMPLE HEADER NAME',
        'html' => '<div>Header
            <span style="font-family: \'comic sans ms\'', sans-
            serif; font-size: 14pt;" xml:lang="example">example</span>

```

```

        <strong>text</strong>
    </div>',
    'text' => 'PLAIN TEXT CONTENT',
    'isHeader' => true
);

print "Adding the header\n";
$write_result = $client->addHeaderFooters(array($headerFooterObject)
    )->return;

// Note we are accessing the results and errors arrays.
// Both of these arrays are returned as part of
// writeResult object.

if ($write_result->errors) {
    print "There was a problem adding the header:\n";
    print_r($write_result->results);
} else {
    print "The header has been added. Id: " . $write_result->results[0]-
>id . "\n";
}

} catch (Exception $e) {
    print "uncaught exception\n";
    print_r($e);
}
?>

```

addToList

The `addToList` function allows you to add one or more contacts to a list. You are limited to adding 5k contacts per call.

Syntax

```
writeResult = bApi.addToList(mailListObject list, contactObject[] contacts);
```

Required and Optional List Object Attributes

Name	Type	Required	Description
id	string	Yes (unless the list name is used)	The unique id assigned to the list. You can obtain the id for a list by calling readLists , or by looking at the footer when viewing the overview page for an individual list in the application.
name	string	Yes (unless name the list id is used)	The internal name of the list.

Required and Optional Contact Object Attributes

Name	Type	Required	Description
id	string	Yes (unless name the contact email is used)	The unique id assigned to the contact. You can obtain the id for a contact by calling readContacts , or by looking at the footer when viewing the overview page for an individual contact in the application. You are limited to adding 5k contacts per call.
email	string	Yes (unless name the contact id is used)	The email address assigned to the contact. The email address can be used to reference a specific contact when using the contact functions.

PHP Code Example

```
<?php
/**
 * This script will incrementally add a contact to a list. The contact will
 * not
 * be dropped from any lists that they have already joined. If you want to
 * add a
 * contact to a list and remove the contact from any existing lists, use
 * the
 * addOrUpdateContacts function. To remove a contact from a list without
 * altering membership to other lists, use the removeFromList function.
 *
 * @copyright Copyright (c) 2018 Oracle + Bronto Software (http://
www.bronto.com)
 */

$client = new SoapClient('https://api.bronto.com/v4?wsdl', array('trace' =>
1,
                                'features' => SOAP_SINGLE_ELEMENT_ARRAYS));

try {
    // Add your API token
    $token = "ADD YOUR API TOKEN";

    print "Logging in\n";
    $sessionId = $client->login(array('apiToken' => $token))->return;

    $session_header = new SoapHeader(
        "http://api.bronto.com/v4",
        'sessionHeader',
        array('sessionId' => $sessionId)
    );
    $client->__setSoapHeaders(array($session_header));

    /**
     * $mailListObject is an array containing the list information.
     * You can pass the list id and/or the list name.
     * The list id is the unique id assigned to the list. You can obtain
```



```

    * the id for a list by calling readLists, or by looking at the footer
    * when viewing the overview page for an individual list in the
application.
    * The list name is the internal name for your list. We recommend
    * using the list id instead of the list name to speed up your API
calls.
    */

    // Example with both id and name included.
    $mailListObject = array("id" => "LIST ID",
                           "name" => "LIST NAME");

    // Example with only list id.
    // $mailListObject = array("id" => "LIST ID");

    // Example with only internal name.
    // $mailListObject = array("name" => "LIST NAME");

    /**
    * $responseObject is an array containing the contact information.
    * The contact API id is a unique identifier assigned to a contact.
    * You can find the contact API id for a contact by looking at the footer
when
    * viewing the overview page for an individual contact in the application.
    * The contact email address can also be used as an id when adding
contact(s).
    */

    // Example using both id and email.
    $responseObject = array("id" => "CONTACT ID",
                           "email" => "CONTACT EMAIL ADDRESS");

    print "Adding the contact(s) to the list\n";

    // Example with id only.
    // $responseObject = array("id" => "CONTACT ID");

    // Example with email only.
    // $responseObject = array("email" => "CONTACT EMAIL ADDRESS");

    // Example with multiple contacts.
    // $contact1 = array("email" => "FIRST CONTACT EMAIL");
    // $contact2 = array("email" => "SECOND CONTACT EMAIL");
    // $contact3 = array("email" => "THIRD CONTACT EMAIL");
    // $responseObject = array($contact1, $contact2, $contact3);

    $write_result = $client->addToList(array('list' =>
$mailListObject, 'contacts' => $responseObject))->return;

    if ($write_result->errors) {
        print "There was a problem adding the contact(s) to the list:\n";
        print_r($write_result->results);
    } else {
        print "The contact(s) have been added.\n";
    }
} catch (Exception $e)
    print "uncaught exception\n";
    print_r($e);
}
?>

```

addLogins

The `addLogins` function allows you to add a new login (aka user) and assign permissions.

Overview

The permissions that you are able to assign to a login depend on what type of account (agency, client account, or professional) you are creating the login for.

Syntax

```
writeResult = bApi.addLogins(loginObject[] logins);
```

Required and Optional List Object Attributes

Name	Type	Required	Description
username	string	Yes	The username assigned to the login.
password	string	Yes	The password assigned to the login.
contactInformation	ContactInformation[]	Yes	An array of the contact information set for the account. Click in the Type column for more information.
permissionAgencyAdmin	boolean	Yes	Gives the login agency administration permissions if you are creating a login for an agency account.
permissionAdmin	boolean	Yes	Gives the login administrative permission if you are creating a login for a client-account or a professional account.
permissionApi	boolean	Yes	Gives the login API permission if you are creating a login for a client-account or a professional account.
permissionUpgrade	boolean	Yes	Gives the login permission to purchase upgrades, such as inbox preview and additional fields. Applicable if you are creating a login for a client-account or professional account.
permissionFatigueOverride	boolean	Yes	Gives the login permission to override any contact frequency caps you have set for them. Applicable if you are creating a login for a client-account or professional account.

Name	Type	Required	Description
permissionMessageCompose	boolean	Yes	Gives the login permission to create messages if you are creating a login for a client-account or a professional account.
permissionMessageDelete	boolean	Yes	Gives the login permission to delete messages if you are creating a login for a client-account or a professional account.
permissionAutomatorCompose	boolean	Yes	Gives the login permission to create automated message rules if you are creating a login for a client-account or a professional account.
permissionListCreateSend	boolean	Yes	Gives the login permission to create and send to lists if you are creating a login for a client-account or a professional account.
permissionListCreate	boolean	Yes	Gives the login permission to create, but not send messages to lists if you are creating a login for a client-account or a professional account.
permissionSegmentCreate	boolean	Yes	Gives the login permission to create segments if you are creating a login for a client-account or a professional account.
permissionFieldCreate	boolean	Yes	Gives the login permission to create fields if you are creating a login for a client-account or a professional account.
permissionFieldReorder	boolean	Yes	Gives the login permission to create messages if you are creating a login for a client-account or a professional account.
permissionSubscriberCreate	boolean	Yes	Gives the login permission to create contacts if you are creating a login for a client-account or a professional account.
permissionSubscriberView	boolean	Yes	Gives the login permission to view contacts if you are creating a login for a client-account or a professional account.

addMessageFolders

The `addMessageFolders` function allows you to add a new message folders to the account. Message folders can be used to group messages.

Syntax

```
writeResult = bApi.addMessageFolders(messageFolderObject[] messageFolders);
```

Required and Optional List Object Attributes

Name	Type	Required	Description
name	string	Yes	The name assigned to the folder.
parentId	string	No; default is the root (top-level) folder	The unique id assigned to the parent folder which contains this folder.

PHP Code Example

```
<?php
/**
 * This script will add a new message folder that is a sub/child
 * folder of an existing message folder. It will then print a
 * visual representation of the folder structure.
 *
 * @copyright Copyright (c) 2011 Bronto Software (http://www.bronto.com)
 */

$client = new SoapClient('https://api.bronto.com/v4?wsdl', array('trace' =>
    1,
    'features' => SOAP_SINGLE_ELEMENT_ARRAYS));

try {
    // Add a valid API token
    $token = "ADD API TOKEN HERE";

    print "logging in\n";
    $sessionId = $client->login(array('apiToken' => $token))->return;

    $session_header = new SoapHeader("http://api.bronto.com/v4",
        'sessionHeader',
        array('sessionId' => $sessionId));
    $client->__setSoapHeaders(array($session_header));

    // Get the ID of the parent folder we want to add
    // a sub/child folder under. We are filtering based
    // on the name of the parent folder.
    // Be sure to replace the value with the name of the
    // parent folder you want an ID for.

    print "Reading message folders that match the given filter\n";

    $filter = array('type' => 'OR',
        'name' => array('operator' => 'EqualTo',
            'value' => 'NAME OF MESSAGE FOLDER'
        )
    );

    $folders = $client->readMessageFolders(array('pageNumber' => 1,
        'filter' => $filter,
```

```

        )
    )->return;
foreach ($folders as $folder) {
    print "Parent folder ID: " . $folder->id . "\n";
}

// Add a nested message folder that is a
// sub/child folder of the existing
// message folder we got an ID for above.
// You may want to replace the name with a
// more relevant message folder name.

$messageFolders = array('name' => 'Child Folder',
                        'parentId' => $folder->id);

print "Adding the child message folder\n";
$write_result = $client->addMessageFolders(array($messageFolders)
    )->return;

if ($write_result->errors) {
    print "There was a problem adding the message folder:\n";
    print_r($write_result->results);
} else {

    // Upon a successful add, we are going to return the ID of
    // the newly created folder. We will also return a
    // visual representation of the message folder structure
    // with the newly created message folder included.

    print "The message folder has been created. Id: " . $write_result-
>results[0]->id . "\n";
    print "Now let's have a look at the folder structure:\n";

    // get all folders
    $filter = array();

    print "reading message folders\n";
    $folders = $client->readMessageFolders(array('pageNumber' => 1,
                                                'filter' => $filter,
                                                )
    )->return;

    // Find the root first
    $rootFolder = null;
    foreach($folders as $folder) {
        if (!$folder->parentId) {
            $rootFolder = $folder;
            break;
        }
    }

    // Start building up a visual tree view of the folders
    $folderStack = array(array('folder' => $rootFolder,
                              'depth' => 0));

    $output = "";
    while (count($folderStack) > 0) {
        $item = array_pop($folderStack);
        $currentFolder = $item['folder'];
        $depth = $item['depth'];
        $indent = str_repeat("    ", $depth);
        $output .= $indent . $currentFolder->name . "\n";
    }
}

```

```

        foreach ($folders as $folder) {
            if ($folder->parentId == $currentFolder->id) {
                array_push($folderStack, array('folder' => $folder, 'depth' =>
$depth + 1));
            }
        }

        // Print out the tree
        print $output . "\n";
    } // End else
} catch (Exception $e) {
    print "uncaught exception\n";
    print_r($e);
}
?>

```

addMessageRules

The `addMessageRules` function allows you to add Automated Message Rules used to trigger automatic email delivery to groups of contacts.

Syntax

```
writeResult = bApi.addMessageRules(messageRuleObject[] messageRules);
```

Required and Optional MessageRule Object Attributes

Name	Type	Required	Description
name	string	Yes	The name assigned to the Automated Message Rule.
type	string	Yes	The type assigned to the Automated Message Rule: <code>activity</code> (behavioral/activity-based), <code>date</code> (date field-based), <code>recurring</code> (recurring based on a particular date or date range), and <code>api</code> (triggered via an API call).
messageId	string	Yes	The unique id of the Message that will be sent via the Automated Message Rule.

PHP Code Example

```

<?php
/**
 * This script will obtain the ID of a message and then create an
 * automated message rule which sends the message whose ID was
 * previously obtained.
 *
 * @copyright Copyright (c) 2011 Bronto Software (http://www.bronto.com)
 */

$client = new SoapClient('https://api.bronto.com/v4?wsdl', array('trace' =>
1,
                        'features' => SOAP_SINGLE_ELEMENT_ARRAYS));

try {
    // Add a valid API token

```

```

$token = "ADD API TOKEN HERE";

print "logging in\n";
$sessionId = $client->login(array('apiToken' => $token))->return;

$session_header = new SoapHeader("http://api.bronto.com/v4",
    'sessionHeader',
    array('sessionId' => $sessionId));
$client->__setSoapHeaders(array($session_header));

// Obtain the ID of the message you want to
// send via the automated message rule.
// Be sure replace the value with the name of the
// message you want to use.

$filter = array('name' => array('operator' => 'EqualTo',
    'value' => 'SOME MESSAGE NAME')
);

print "Reading all matching messages\n";

$messages = $client->readMessages(array('pageNumber' => 1,
    'includeContent' => false,
    'filter' => $filter))->return;

// return is an array of objects found that match the given
// filter. If nothing is found, return will be NULL and 0
// is returned for count(), hence the check for > 0.
// Ideally, you should make your filter specific enough so
// that only the message you want is returned, however, in the
// event that it is not, we'll get the first item.

if (count($messages) > 0) {
    print "The message Id is: " . $messages[0]->id . "\n";
} else {
    exit("No messages were found that match the given filter.\n");
}

// Add an automated message rule which will send
// the message whose ID we obtained above.

$messageRules = array('name' => 'Example AMR',
    'type' => 'api',
    'messageId' => $messages[0]->id
);

print "Adding the automated message rule\n";
$write_result = $client->addMessageRules(array($messageRules)
    )->return;

if ($write_result->errors) {
    print "There was a problem adding the automated message rule:\n";
    print_r($write_result->results);
} else {
    print "The automated message rule has been created. Id: " .
$write_result->results[0]->id . "\n";
}

} catch (Exception $e) {
    print "uncaught exception\n";
    print_r($e);
}

```

```
}
?>
```

addMessages

The addMessages function allows you to add new messages to your account.

Overview

Currently the API does not support the structure of email message editor messages. Any messages you add with addMessages can only be opened in the platform using the HTML or WYSIWYG message editor.



Note: When you insert dynamic links in messages via the API, Bronto will only track up to 30 dynamic links for single send messages and up to 10 for bulk messages.

Syntax

```
writeResult = bApi.addMessages(messageObject[] messages);
```

Required and Optional Message Object Attributes

Name	Type	Required	Description
name	string	Yes	the name of this message
messageFolderId	string	No; default is the root (top-level) folder	the id of the folder containing this message
content	MessageContentObject[]	No	array of contents for this message
deeplink	>MessageContentObject[]	No	part of the array, determines if a link is a deep link.

PHP Code Example

```
<?php
/* This script will add an HTML message with some basic HTML content.
   @copyright Copyright (c) 2018 Bronto Software (http://www.bronto.com) */

$client = new SoapClient('https://api.bronto.com/v4?wsdl', array('trace' =>
    1, 'features' => SOAP_SINGLE_ELEMENT_ARRAYS));
try {
    // Add in a valid API token
    $token = "ADD API TOKEN HERE";
    print "logging in\n";
    $sessionId = $client->
        login(array('apiToken' => $token))->return;
    $session_header = new SoapHeader("http://api.bronto.com/
v4", 'sessionHeader', array('sessionId' => $sessionId));
    $client->__setSoapHeaders(array($session_header));

    // Be sure to replace the example subject and content
    $messageContentObject = array('type' => 'html', 'subject' => 'Example
Subject Line', 'content' => '<h1> Some HTML can go here</h1><p>Some more
can go here.</p>' );

    // Give the message a valid name
    $message = array('name' => 'Example Message Name', 'content' =>
    $messageContentObject );
    print "Adding the message\n";
```



```

    $write_result = $client>;addMessages(array($message) )->return; if
    ($write_result->errors) { print "There was a problem adding the message:
    \n";
    print_r($write_result->results);
    } else {
    print "The message has been created. Id: " . $write_result->results[0]-
    >id . "\n";
    }
    } catch (Exception $e) { print "uncaught exception\n";
    print_r($e);
    }
    ?>

```

addOrUpdateContacts

The `addOrUpdateContacts` function allows you to add a new contact, or update an existing contact.

Overview

You can also add or update data associated with the contact, such as field data and list membership.

Syntax

```
writeResult writeR = bApi.addOrUpdateContacts(contactObject[] contacts);
```

Required and Optional Message Object Attributes

Name	Type	Required	Description
id	string	No if the email or mobileNumber is provided.	The id can be used to reference a specific contact to update, but the id itself can not be updated.
email	string	No if the id or mobileNumber is provided.	The email address stored for the contact. The email address can be used to reference a specific contact and can also be updated.
mobileNumber	string	No if the email or id is provided.	The mobile number stored for the contact. The mobile number can be used to reference a specific contact and can also be updated. A valid country code must be included when adding or updating a mobile number for a contact.
status	string	No; default for adding is onboarding	The status of the contact. For adding, the status is automatically set to onboarding, unless you specifically set the status as unconfirmed or transactional. If you want to update an existing contact's status, you must use the updateContacts function.
msgPref	string	No; default is html	The message preference for the contact. A contact can have a message preference of text or html. Only applies to adds. The message preference is ignored in the case of an update

Name	Type	Required	Description
source	string	No; default is api	The source or where the contact came from. The source can be manual, import, api, webform, or sforcereport (salesforce report).
customSource	string	No; default is empty	A source you define that states where the contact came from.
listIds	string, array. Use an array for multiple ids	No	<p>The lists (referenced by ID) that the contact belongs to. You obtain listIds by calling the <code>readLists</code> function.</p> <p>The lists you set in this call are absolute, not incremental, to lists the contact may already be on. This means contacts are removed from any list(s) not specified in this call and will only be added to lists you specify in this call.</p> <p>However, this cannot be used to pass an empty list of ids in order to remove contacts from all lists that it belongs to. When the API recognizes that the list is completely empty the listId portion of the call is ignored</p> <p>If you want to</p> <ul style="list-style-type: none"> Incrementally add a contact to a list without affecting their membership on other lists, use the addToList function. Incrementally remove a contact from a list, use the removeFromList function. <p>If you want to use this call to incrementally add the contact to a new list and retain their current list membership, you'll need to</p> <ul style="list-style-type: none"> Call readContacts. Obtain the ids for the lists the contact is currently a member of. Pass in those ids along with the new list ids when calling <code>addOrUpdateContacts</code>.
fields	contactField[]	No	An array of the fields and corresponding field data associated with the contact.
SMSKeywordIDs	string, array. Use an array for multiple ids	No	An array of the SMS keyword ids you want to subscribe the contact to.

PHP Code Example

```
<?php
/**
```

```

* This script will add a contact if the contacts is new, or update the
contact's
* information if they already exist.
*
* @copyright Copyright (c) 2018 Bronto Software (http://www.bronto.com)
*/

$client = new SoapClient('https://api.bronto.com/v4?wsdl', array('trace' =>
1,
                        'features' => SOAP_SINGLE_ELEMENT_ARRAYS));

try {
    // Add in a valid API token
    $token = "ADD YOUR TOKEN HERE";

    print "logging in\n";
    $sessionId = $client->login(array('apiToken' => $token))->return;

    $session_header = new SoapHeader("http://api.bronto.com/v4",
                                    'sessionHeader',
                                    array('sessionId' => $sessionId));
    $client->__setSoapHeaders(array($session_header));

    // Replace SOME CONTENT with a string. We assume here
    // the field is storing a string. The value you pass in
    // should match the type set for the field.
    // Replace SOME FIELD ID with a valid field ID. Field IDs
    // can be obtained by calling readFields. Field IDs are also
    // available in the footer when viewing an individual field in
    // the UI.
    $field1 = array('fieldId' => 'SOME FIELD ID',
                   'content' => 'SOME CONTENT');
    $field2 = array('fieldId' => 'SOME FIELD ID',
                   'content' => 'SOME CONTENT');

    // Note: The lists you set in this call will be absolute, not
    // incremental, to lists the contact may already be on. The contact
    // will be removed from any list(s) not specified in this call and
    // will only be added to lists you specify in this call. If your intent
    // is to incrementally add a contact to a list without affecting their
    // membership on other lists, use the addToList function. If you want to
    // incrementally remove a contact from a list, use the removeFromList
    // function. If you want to use this call to incrementally add the
    contact
    // to a new list and retain their current list membership, you'll need to
    // call readContacts, obtain the ids for the lists the contact is
    currently
    // a member of, and pass in those ids along with the new list ids when
    // calling updateContacts.
    $contacts = array(
        'email' => 'EMAIL ADDRESS',
        'listIds' => 'LIST ID',
        'fields' => array(
            $field1,
            $field2
        ),
        'customSource' => 'source'
    );

    print "Adding contact with the following attributes\n";
    $write_result = $client->addOrUpdateContacts(array($contacts)
    )->return;
}

```

```

    if ($write_result->errors) {
        print "There was a problem adding or updating the contact:\n";
        print_r($write_result->results);
    } elseif ($write_result->results[0]->isNew == true) {
        print "The contact has been added. Contact Id: " . $write_result->results[0]->id . "\n";
    } else {
        print "The contact's information has been updated. Contact Id: " . $write_result->results[0]->id . "\n";
    }
} catch (Exception $e) {
    print "uncaught exception\n";
    print_r($e);
}

```

Python Code Example

```

import sys
import logging
from suds.client import Client
from suds import WebFault

# This example script will login in to the API, add
# a contact, add them to a list, and add some field
# data if the contact is new. If the contact already
# exists, they will be added to the list and their
# field data will be updated

# BE SURE TO REPLACE ALL PLACEHOLDER TEXT

# Tested with Python 2.6.1 and suds soap library version 0.4

# See suds home page:
# <a title="https://fedorahosted.org/suds/" href="https://fedorahosted.org/suds/">https://fedorahosted.org/suds/</a>

# Bronto API WSDL
BRONTO_WSDL = 'https://api.bronto.com/v4?wsdl'

# Start up basic logging
logging.basicConfig()

# Replace the placeholder text with a valid
# API token
TOKEN = "ADD API TOKEN"

# Login using the token to obtain a session ID
bApi = Client( BRONTO_WSDL )

try:
    session_id = bApi.service.login(TOKEN)
# Just exit if something goes wrong
except WebFault, e:
    print '\nERROR MESSAGE: '
    print e
    sys.exit()

```

```

# Set up the soap headers using the
# session_id obtained from login()
session_header = bApi.factory.create("sessionHeader")
session_header.sessionId = session_id
bApi.set_options(soapheaders=session_header)

# Set up the contactField objects
# Replace SOME CONTENT with a string. We assume here
# the field is storing a string. The value you pass in
# should match the type set for the field.
# Replace SOME FIELD ID with a valid field ID. Field IDs
# can be obtained by calling readFields. Field IDs are also
# available in the footer when viewing an individual field in
# the UI.
field1 = bApi.factory.create('contactField')
field1.fieldId = "SOME FIELD ID"
field1.content = "SOME CONTENT"

field2 = bApi.factory.create('contactField')
field2.fieldId = "SOME FIELD ID"
field2.content = "SOME CONTENT"

# Adding a contact, assigning them to a list,
# and adding some field data.
# Be sure to replace the placeholder text with a
# real email address and list ID!

contact = bApi.factory.create('contactObject')
contact.email = 'some_email@example.com'
contact.fields = [field1, field2]

# contact.listIds = 'SOME LIST ID'

# Note: The lists you set in this call will be absolute, not
# incremental, to lists the contact may already be on. The contact
# will be removed from any list(s) not specified in this call and
# will only be added to lists you specify in this call. If your intent
# is to incrementally add a contact to a list without affecting their
# membership on other lists, use the addToList function. If you want to
# incrementally remove a contact from a list, use the removeFromList
# function. If you want to use this call to incrementally add the contact
# to a new list and retain their current list membership, you'll need to
# call readContacts, obtain the ids for the lists the contact is currently
# a member of, and pass in those ids along with the new list ids when
# calling updateContacts.

try:
    add_contact = bApi.service.addOrUpdateContacts(contact)
    if add_contact.results[0].isError == True:
        print 'There was an error with your request:'
        print add_contact.results[0]
        sys.exit()
    elif add_contact.results[0].isNew == True:
        print 'A contact has been added with the id: ' +
add_contact.results[0].id
    else:
        print 'The following contact has been updated: ' +
add_contact.results[0].id
except WebFault, e:
    print '\nERROR MESSAGE:'
    print e
    sys.exit()

```

addOrUpdateContactsIncremental

The `addOrUpdateContactsIncremental` function allows you to add a new contact or update data associated with an existing contact without replacing the existing contact's list membership.

Overview


With the exception of how list membership is managed, this call works exactly like the `addOrUpdateContacts` function.

Syntax

```
writeResult = bApi.addOrUpdateContacts(contactObject[] contacts);
```

Required and Optional Message Object Attributes

Name	Type	Required	Description
id	string	No if the email or mobileNumber is provided.	The id can be used to reference a specific contact to update, but the id itself can not be updated.
email	string	No if the id or mobileNumber is provided.	The email address stored for the contact. The email address can be used to reference a specific contact and can also be updated.
mobileNumber	string	No if the email or id is provided.	The mobile number stored for the contact. The mobile number can be used to reference a specific contact and can also be updated. A valid country code must be included when adding or updating a mobile number for a contact.
status	string	No; default for adding is onboarding	The status of the contact. For adding, the status is automatically set to onboarding, unless you specifically set the status as unconfirmed or transactional. If you want to update an existing contact's status, you must use the updateContacts function.
messagePreference	string	No; default is html	The message preference for the contact. A contact can have a message preference of text or html. Only applies to adds. The message preference is ignored in the case of an update
source	string	No; default is api	The source or where the contact came from. The source can manual, import, api, webform, or sfcoreport (salesforce report).
customSource	string	No; default is empty	A source you define that states where the contact came from.

Name	Type	Required	Description
listIds	string, array. Use an array for multiple ids	No	The lists (referenced by ID) that the contact belongs to. You obtain listIds by calling the readLists function.  Note: The lists you set in this call are incremental to lists the contact may already be on. This means contact will remain on any list(s) not specified in this call and is also added to lists you specify in this call. If you want to incrementally remove a contact from a list, use the removeFromList function.
fields	contactField[]	No	An array of the fields and corresponding field data associated with the contact.
SMSKeywordIDs	string, array. Use an array for multiple ids	No	An array of the SMS keyword ids you want to subscribe the contact to.

PHP Code Example

```
<?php /** * This script will add a contact if the contacts is new, or
update the contact's * information if they already exist. * * @copyright
Copyright (c) 2011 Bronto Software (http://www.bronto.com) */ $client = new
SoapClient('https://api.bronto.com/v4?wsdl', array('trace' => 1,
'features' => SOAP_SINGLE_ELEMENT_ARRAYS));

try {
    // Add in a valid API token
    $token = "ADD YOUR TOKEN HERE";

    print "logging in\n";
    $sessionId = $client->login(array('apiToken' => $token))->return;

    $session_header = new SoapHeader("http://api.bronto.com/v4",
        'sessionHeader',
        array('sessionId' => $sessionId));
    $client->__setSoapHeaders(array($session_header));

    // Replace SOME CONTENT with a string. We assume here
    // the field is storing a string. The value you pass in
    // should match the type set for the field.
    // Replace SOME FIELD ID with a valid field ID. Field IDs
    // can be obtained by calling readFields. Field IDs are also
    // available in the footer when viewing an individual field in
    // the UI.
    $field1 = array('fieldId' => 'SOME FIELD ID',
        'content' => 'SOME CONTENT');
    $field2 = array('fieldId' => 'SOME FIELD ID',
        'content' => 'SOME CONTENT');

    // Note: The lists you set in this call will be incremental to the lists
```

```
// the contact may already be on. The contact will not be removed from any
// list(s). To incrementally remove a contact from a list,
// use the removeFromList function.
$contacts = array('email' => 'some_contact@example.com',
                  'listIds' => 'ADD IN A LIST ID',
                  'fields' => array($field1, $field2)
                  );

print "Adding contact with the following attributes\n";
$write_result = $client->addOrUpdateContacts(array($contacts)
                                             )->return;

if ($write_result->errors) {
    print "There was a problem adding or updating the contact:\n";
    print_r($write_result->results);
} elseif ($write_result->results[0]->isNew == true) {
    print "The contact has been added. Contact Id: " . $write_result-
>results[0]->id . "\n";
} else {
    print "The contact's information has been updated. Contact Id: " .
$write_result->results[0]->id . "\n";
}

} catch (Exception $e) {
    print "uncaught exception\n";
    print_r($e);
}
```

Python Code Example

```
import sys
import logging
from suds.client import Client
from suds import WebFault

# This example script will login in to the API, add
# a contact, add them to a list, and add some field
# data if the contact is new. If the contact already
# exists, they will be added to the list and their
# field data will be updated

# BE SURE TO REPLACE ALL PLACEHOLDER TEXT

# Tested with Python 2.6.1 and suds soap library version 0.4

# See suds home page:
# <a title="https://fedorahosted.org/suds/" href="https://fedorahosted.org/
suds/">https://fedorahosted.org/suds/</a>

# Bronto API WSDL
BRONTO_WSDL = 'https://api.bronto.com/v4?wsdl'

# Start up basic logging
logging.basicConfig()

# Replace the placeholder text with a valid
# API token
TOKEN = "ADD API TOKEN"
```



```

# Login using the token to obtain a session ID
bApi = Client( BRONTO_WSDL )

try:
    session_id = bApi.service.login(TOKEN)
# Just exit if something goes wrong
except WebFault, e:
    print '\nERROR MESSAGE: '
    print e
    sys.exit()

# Set up the soap headers using the
# session_id obtained from login()
session_header = bApi.factory.create("sessionHeader")
session_header.sessionId = session_id
bApi.set_options(soapheaders=session_header)

# Set up the contactField objects
# Replace SOME CONTENT with a string. We assume here
# the field is storing a string. The value you pass in
# should match the type set for the field.
# Replace SOME FIELD ID with a valid field ID. Field IDs
# can be obtained by calling readFields. Field IDs are also
# available in the footer when viewing an individual field in
# the UI.
field1 = bApi.factory.create('contactField')
field1.fieldId = "SOME FIELD ID"
field1.content = "SOME CONTENT"

field2 = bApi.factory.create('contactField')
field2.fieldId = "SOME FIELD ID"
field2.content = "SOME CONTENT"

# Adding a contact, assigning them to a list,
# and adding some field data.
# Be sure to replace the placeholder text with a
# real email address and list ID!

contact = bApi.factory.create('contactObject')
contact.email = 'some_email@example.com'
contact.fields = [field1, field2]

# contact.listIds = 'SOME LIST ID'

# Note: The lists you set in this call will be incremental to
# the lists the contact may already be on. The contact will not
# be removed from any list(s). To incrementally remove a contact
# from a list, use the removeFromList function.

try:
    add_contact = bApi.service.addOrUpdateContacts(contact)
    if add_contact.results[0].isError == True:
        print 'There was an error with your request: '
        print add_contact.results[0]
        sys.exit()
    elif add_contact.results[0].isNew == True:
        print 'A contact has been added with the id: ' +
add_contact.results[0].id
    else:

```

```

        print 'The following contact has been updated: ' +
add_contact.results[0].id
except WebFault, e:
    print '\nERROR MESSAGE:'
    print e
    sys.exit()

```

addOrUpdateDeliveryGroup

The addOrUpdateDeliveryGroup function allows you to add a delivery group if it is new, or update the delivery group if it already exists.

Syntax

```

writeResult = bApi.addOrUpdateDeliveryGroup(deliveryGroupObject[]
deliveryGroup);

```

Required and Optional Message Object Attributes

Name	Type	Required	Description
id	string	Required if no name is given.	The unique id for the delivery group.
name	string	Required if no id is given.	The name associated with the delivery group.
deliveryIds[]	string	No	An array of the delivery ids you want to associate with the delivery group.
messageRuleIds[]	string	No	An array of the automated message rule ids you want to associate with the delivery group.
messageIds[]	string	No	An array of the message ids you want to associate with the delivery group.

PHP Code Example

```

<?php
/**
 * This example will add the delivery group if it is new, and update
 * the delivery group if it already exists.
 */

$client = new SoapClient('https://api.bronto.com/v4?wsdl', array('trace' =>
1,
                                'features' => SOAP_SINGLE_ELEMENT_ARRAYS));
setlocale(LC_ALL, 'en_US');

try {
    // Add in a valid API token
    $token = "ADD API TOKEN HERE";

    print "logging in\n";
    $sessionId = $client->login(array('apiToken' => $token))->return;
}

```

```

$session_header = new SoapHeader("http://api.bronto.com/v4",
    'sessionHeader',
    array('sessionId' => $sessionId)
);
$client->__setSoapHeaders(array($session_header));

// Be sure to replace the generic values below

$deliveryIds = array('SOME DELIVERY ID');
$deliveryGroupObject = array('name' => 'SOME DELIVERY GROUP'
    'deliveryIds' => $deliveryIds
);

$write_result = $client-
>addOrUpdateDeliveryGroup(array($deliveryGroupObject))->return;

    if ($write_result->errors) {
        print "There was a problem adding or updating the delivery group:
\n";
        print_r($write_result->results);
    } elseif ($write_result->results[0]->isNew == true) {
        print "The delivery group has been created.  Id: " . $write_result-
>results[0]->id . "\n";
    } else {
        print "The delivery group has been updated.  Id: " . $write_result-
>results[0]->id . "\n";
    }

} catch (Exception $e) {
    print "uncaught exception\n";
    print_r($e);
}
?>

```

addOrUpdateOrders

The addOrUpdateOrders call will create a new order if the order id does not exist or update an existing order.

Overview

It will update the order via the following logic:

- The orderDate and products will be updated if they are not null. When updating the products array, the entire array is replaced. That means you are expected to pass in the current contents of the order and not only the products that have changed. Please reference the Notes in the Description column below for more details.
- The email, contactId, and tid are all capable of associating an order with a contact.
- The tid allows you to associate an order with a delivery and a contact. The tid is taken from the cookie that is generated when clicking a link that redirects through Bronto.







Important: If the order already exists, the email, contactID, and tid parameters will be ignored.

If you wish to change the items an order is associated with (contact or delivery), you can call deleteOrders to delete the order, and then call addOrUpdateOrders to re-add the order with the proper associations.

Syntax

```
writeResult = bApi.addOrUpdateOrders(orderObject[] orders);
```

Required and Optional Order Object Attributes

Name	Type	Required	Description
id	string	Yes	Unique ID of the order. If the order <code>id</code> already exists within Bronto, the order will be updated. However, if the order <code>id</code> does not exist, a new order will be added.
email	string	No	Email address of the person placing the order. If a corresponding contact does not exist then a new one will be created with the status of <code>transactional</code> . If the order <code>id</code> already exists, then this field is ignored and not updated.
contactId	string	No	The unique ID assigned to the contact placing the order. If the order <code>id</code> already exists, then this field is ignored and not updated.
products	productObject[]	No	<p>Array of the products contained in this order.</p> <p> Note: You are limited to 500 product objects per order.</p> <p> Note: This is not required, so you can add an order and come back later to fill in the product details. When the order <code>id</code> does already exist:</p> <ul style="list-style-type: none"> • If this is not provided at all (as in null) then it is not replaced. • If an empty array is provided (as in not null but the array has no values), then all products are removed from the order. • If an array with values is provided, then all products in the order are replaced with this new set.
orderDate	dateTime	No	<p>Date and time of the order.</p> <p> Note: If no value is provided, the system will timestamp the record. You can (and should) specify a timezone offset if you do not want the system to assume you are providing a time in UTC (Coordinated Universal Time / Greenwich Mean Time). For the Eastern Time Zone on Daylight Savings Time, this would be:</p> <p><i>YYYY-MM-DDTHH:MM:SS-04:00</i></p>
tid	string	No	<p>A unique id that associates an order with a specific contact and delivery. Since this ties the order back to a delivery, it is then classified as a conversion for that specific delivery. T</p> <p> Note: The <code>tid</code> can only be pulled from the Bronto tracking cookie, which requires that Conversion Tracking be enabled within the Bronto application. If the order <code>id</code> already exists, then this field is ignored and not updated.</p>

PHP Code Example

```

<?php
/**
 * This script will add an order which contains two products.
 *
 * @copyright Copyright (c) 2012 Bronto Software (http://www.bronto.com)
 */

$client = new SoapClient('https://api.bronto.com/v4?wsdl', array('trace' =>
    1,
                                                                    'features'
=> SOAP_SINGLE_ELEMENT_ARRAYS));

try {
    $token = "ADD TOKEN HERE";

    print "logging in\n";
    $sessionId = $client->login(array('apiToken' => $token))->return;

    $session_header = new SoapHeader("http://api.bronto.com/v4",
                                      'sessionHeader',
                                      array('sessionId' => $sessionId));
    $client->__setSoapHeaders(array($session_header));

    // Both the products below contain generic data. Be sure
    // to replace this data with real product data.
    $product1 = array(
        'id' => '78923',
        'sku' => '23424',
        'name' => 'Soccer Ball',
        'description' => 'Blue Soccer Ball',
        'category' => 'Sporting Goods',
        'quantity' => 3,
        'price' => 45.95
    );

    $product2 = array(
        'id' => '56735',
        'sku' => '6544',
        'name' => 'Guards',
        'description' => 'Shin guards',
        'category' => 'Sporting Goods',
        'quantity' => 1,
        'price' => 25.95
    );

    // If you want to use the tid, uncomment the following foreach and
    // and the tid item in the orderObject array. Make sure the code
    // is used in a location that has access to $_COOKIE variable which
    // contains the tid you want to use in this call.

    //foreach ($_COOKIE as $cookie=>$value) {
    //    if (strpos($cookie, "tid_") !== false) {
    //        $tid = $value;
    //    }
    //}

    // Add an order which contains two products.
    // Be sure to replace the generic values below.
    $orderObject = array(
        'id' => '2341234',

```

```

        'email' => 'CONTACT EMAIL ADDRESS',
        'products' => array($product1, $product2),
        //'tid' => $tid,
        'orderDate' => date('c'),
    );

    print "Adding order with the following attributes\n";
    $write_result = $client->addOrUpdateOrders(array($orderObject)
                                              )->return;

    if ($write_result->errors) {
        print "There was a problem adding or updating the order:\n";
        print_r($write_result->results);
    } elseif ($write_result->results[0]->isNew == true) {
        print "The order has been added. Id: " . $write_result->results[0]-
>id . "\n";
    } else {
        print "The order information has been updated. Id: " .
        $write_result->results[0]->id . "\n";
    }

} catch (Exception $e) {
    print "uncaught exception\n";
    print_r($e);
}

?>

```

Python Code Example

```

import sys
import logging
from suds.client import Client
from suds import WebFault
from datetime import datetime

"""
This example script will login to the API and add
an order associated with a contact. The order contains
two products.

BE SURE TO REPLACE ALL PLACEHOLDER TEXT

Tested with Python 2.6.1 and suds soap library version 0.4

See suds home page:
https://fedorahosted.org/suds/

"""

# Bronto API WSDL
BRONTO_WSDL = 'https://api.bronto.com/v4?wsdl'

# Start up basic logging
logging.basicConfig()

# Replace the placeholder text with a valid
# API token
TOKEN = "ADD API TOKEN"

```

```

# Login using the token to obtain a session ID
bApi = Client( BRONTO_WSDL )

try:
    session_id = bApi.service.login(TOKEN)
# Just exit if something goes wrong
except WebFault, e:
    print '\nERROR MESSAGE: '
    print e
    sys.exit()

# Set up the soap headers using the
# session_id obtained from login()
session_header = bApi.factory.create("sessionHeader")
session_header.sessionId = session_id
bApi.set_options(soapheaders=session_header)

product1 = bApi.factory.create('productObject')
product1.id = "324223"
product1.sku = "324223"
product1.name = "Baseball"
product1.description = "Pro Baseball"
product1.category = "Sporting Goods"
product1.quantity = 2
product1.price = 10.99

product2 = bApi.factory.create('productObject')
product2.id = "56756"
product2.sku = "56756"
product2.name = "Glove"
product2.description = "Catcher's glove"
product2.category = "Sporting Goods"
product2.quantity = 1
product2.price = 75.99

# Adding an order which contains two products
# Be sure to replace the placeholder data with real
# order data

order = bApi.factory.create('orderObject')
order.id = '9847623'
order.email = 'CONTACT EMAIL ADDRESS'
order.products = [product1, product2]
order.orderDate = datetime.now()

try:
    add_order = bApi.service.addOrUpdateOrders(order)
    if add_order.results[0].isError == True:
        print 'There was an error with your request: '
        print add_order.results[0]
        sys.exit()
    elif add_order.results[0].isNew == True:
        print 'An order has been added with the id: ' +
add_order.results[0].id
    else:
        print 'The following order has been updated: ' +
add_order.results[0].id
except WebFault, e:
    print '\nERROR MESSAGE: '

```

```
print e
sys.exit()
```

addSMSDeliveries

The `addSMSDeliveries` function allows you to add new SMS deliveries.

Syntax

```
writeResult = bApi.addSMSDeliveries(smsDeliveryObject[] deliveries);
```

Required and Optional Delivery Attributes

Name	Type	Required	Description
start	dateTime	Yes	The date the SMS delivery is scheduled to be sent. You can (and should) specify a timezone offset if you do not want the system to assume you are providing a time in UTC (Universal Coordinated Time / Greenwich Mean Time). For the Eastern Time Zone on Daylight Savings Time, this would be YYYY-MM-DDTHH:MM:SS-04:00.
messageId	string	Yes	The id of the SMS message associated with the delivery. You must reference the SMS message you want to use in the SMS delivery by ID. The <code>messageId</code> can be obtained in the application UI. It will appear in the bottom right corner of the screen when viewing an individual SMS message.
recipients	deliveryRecipientObject[]	No	An array of delvieryRecipientObjects that specify the contact(s) and or keyword(s) the SMS delivery will be sent to
fields	smsMessageFieldObject[]	Yes	If you are using API message tags in your SMS message, you can use the smsMessageFieldObject to specify which content to pass in.

PHP Code Example

```
<?php
/* This example will create an SMS delivery and send it to an entire
keyword. This example * also demonstrates how to use the fields attribute
to pass in data that will replace * API message tags contained in the body
of the SMS message. * Be sure to replace the placeholder text below with
real values. */

$client = new SoapClient('https://api.bronto.com/v4?wsdl', array('trace' =>
1, 'features' => SOAP_SINGLE_ELEMENT_ARRAYS)); setlocale(LC_ALL, 'en_US');

try {
    $token = "ADD YOUR API TOKEN HERE";
    print "logging in\n";
    $sessionId = $client->login(array('apiToken' => $token))->return;
    $session_header = new SoapHeader("http://api.bronto.com/
v4", 'sessionHeader', array('sessionId' => $sessionId));

    $client->__setSoapHeaders(array($session_header)); // Make delivery
start timestamp
    $now = date('c'); // Used to replace API message tags in your SMS message
with the // value you pass in for 'content'.

    $smsMessageFieldObject = array('name' => 'SOME API MESSAGE TAG
NAME', 'content' => 'SOME EXAMPLE CONTENT');
    $delivery = array('start' => $now, 'messageId' => 'ADD SMS
MESSAGE ID HERE', 'keywords' => 'ADD KEYWORD ID HERE', 'fields' =>
$smsMessageFieldObject );
    $write_result = $client->addSMSDeliveries(array($delivery))->return;

    if ($write_result->errors) {
        print "There was a problem adding the SMS delivery:\n";
        print_r($write_result->results);
    }
    else {
        print "The SMS delivery has been successfully created.\n"; }
    }
    catch (Exception $e) {
        print "uncaught exception\n";
        print_r($e);
    }
}

?>
```

Python Code Example

```
import sys
import logging
from suds.client import Client
from suds import WebFault
from datetime import datetime

# This example script will login to the API, and schedule an
# SMS delivery. This example demonstrates how to send to a specific
# contact. It also demonstrates how to
# pass in content to replace API message tags contained in the SMS message.

# BE SURE TO REPLACE ALL PLACEHOLDER TEXT

# Tested with Python 2.6.1 and suds soap library version 0.4
```

```

# See suds home page:
# https://fedorahosted.org/suds/

# Bronto API WSDL
BRONTO_WSDL = 'https://api.bronto.com/v4?wsdl'

# Start up basic logging
logging.basicConfig()

# Replace the placeholder text with a valid
# API token
TOKEN = "ADD API TOKEN HERE"

# Login using the token to obtain a session ID
bApi = Client( BRONTO_WSDL )

try:
    session_id = bApi.service.login(TOKEN)
    # Just exit if something goes wrong
except WebFault, e:
    print '\nERROR MESSAGE: '
    print e
    sys.exit()

# Set up the soap headers using the
# session_id obtained from login()
session_header = bApi.factory.create("sessionHeader")
session_header.sessionId = session_id
bApi.set_options(soapheaders=session_header)

# Get the current date and time so we can send the
# SMS message now
sendtime = datetime.now()

# Set up the smsDeliveryObject
# Be sure to replace the placeholder text below
add_sms_delivery = bApi.factory.create('smsDeliveryObject')
add_sms_delivery.start = sendtime
add_sms_delivery.messageId = 'SMS MESSAGE ID'

# Set up the deliveryRecipientObject object. This allows you
# to send to specific contacts and or keywords. Multiple
# contacts can be sent to by passing in an array of contactIds.
deliveryRecipientObject = bApi.factory.create('deliveryRecipientObject')
deliveryRecipientObject.type = 'contact'
deliveryRecipientObject.id = 'CONTACT ID'
add_sms_delivery.recipients = deliveryRecipientObject

# set up the smsMessageFieldObject. This allows you to pass
# in content that will replace API message tags in the SMS
# message.
smsMessageFieldObject = bApi.factory.create('smsMessageFieldObject')
smsMessageFieldObject.name = 'API MESSAGE TAG NAME'
smsMessageFieldObject.content = 'SOME CONTENT TO PASS IN'
add_sms_delivery.fields = smsMessageFieldObject

try:
    delivery = bApi.service.addSMSDeliveries(add_sms_delivery)
    if delivery.results[0].isError == True:

```

```

        print 'There was an error with your request:'
        print delivery.results[0]
        sys.exit()
    else:
        print 'The SMS delivery has been sent.'
except WebFault, e:
    print '\nERROR MESSAGE:'
    print e
    sys.exit()

```

addSMSKeywords

The addSMSKeywords function allows you to add an SMS keyword.

Overview

If you want to add contacts to an SMS keyword, use addToSMSKeyword. If you want to remove contacts from an SMS keyword, use removeFromSMSKeyword.

Syntax

```
writeResult = bApi.addSMSKeywords(smsKeywordObject[] keyword);
```

Required and Optional Delivery Attributes

Name	Type	Required	Description
name	string	Yes	The name assigned to the SMS keyword.
description	string	No	The description provided for the SMS keyword.
frequencyCap	long	No	The frequency cap represents the maximum number of SMS messages you can send to a person subscribed to the keyword each month. Best practice suggests you set the frequency cap to 30 or less per month. Valid values are 1-30.

Name	Type	Required	Description
confirmationMessage	string	No	The confirmation message set for the SMS keyword. The confirmation message is sent when a user confirms their subscription to the SMS keyword. The text “ <i>Txt STOP to <XXXXX> to end, HELP for info. <XX>msg/mo. Msg&Data rate may apply.</i> ” will be appended to your message. The confirmationMessage can be a maximum of 83 characters.
messageContent	string	No	The message content set for the SMS keyword. The messageContent will be sent each time a contact texts into the keyword or replies to a message from the keyword. The messageContent can be a maximum of 160 characters.

Name	Type	Required	Description
keywordType	string	No	<p>The type set for the SMS keyword. Valid values are:</p> <ul style="list-style-type: none"> • <code>basic</code> – Basic keywords are non-subscription keywords meant for individual transactions. With basic keywords, a person texts into a keyword and a response is sent back. The interaction ends there. The recipient is not added to a list because they have not agreed to receive future marketing messages from you. • <code>subscription</code> – Subscription based keywords require a person to choose to receive SMS messages from you by texting into a given keyword. Contacts who subscribe to a subscription based keyword will be added to a list so that you can send SMS messages to them in the future. • <code>text2join</code> – <code>text2join</code> keywords allows you to grow your list by providing a way for potential contacts to text their email address in and automatically be added to one of your lists.

addSMSMessages

The `addSMSMessages` function allows you to add new SMS messages to your account.

Syntax

```
writeResult = bApi.addSMSMessages(smsMessageObject[] messages);
```

Required and Optional Message Object Attributes

Name	Type	Required	Description
name	string	Yes	The name of the SMS message.
messageFolderId	string	No; default is the root (top-level) folder	The id of a folder to add the SMS message to.
shortenUrls	boolean	Yes	Set to <code>true</code> if the SMS message should use shortened URLs. In addition to shortening URLs, which allows for more content, this setting also allows the application to track the URLs used in the message. URLs will be shortened when the message is sent. The character count takes this into account, and thus represents the count for the message as if the URLs were shortened.
content	string	Yes	The content for the SMS message. SMS messages are limited to 160 characters. The following text must be included somewhere in the body of the SMS message: Text STOP to end

addToDeliveryGroup

The `addToDeliveryGroup` function allows you to add messages, automated message rules, or deliveries to an existing delivery group.

Syntax

```
writeResult = bApi.addToDeliveryGroup(deliveryGroupObject[] deliveryGroup);
```

Required and Optional Delivery Attributes

Name	Type	Required	Description
deliveryGroup	string	Yes	The id associated with the delivery group you want to add items to.
deliveryIds[]	string	No	An array of the delivery ids you want to add to the delivery group.

Name	Type	Required	Description
messageRuleIds[]	string	No	An array of the automated message rule ids you want to add to the delivery group.
messageIds[]	string	No	An array of the message ids you want to add to the delivery group.

PHP Code Example

```

<?php
/**
 * This script will add a delivery, message, and automated message rule
 * to a delivery group.
 *
 * @copyright Copyright (c) 2011 Bronto Software (http://www.bronto.com)
 */

$client = new SoapClient('https://api.bronto.com/v4?wsdl', array('trace' =>
    1,
                                'features' => SOAP_SINGLE_ELEMENT_ARRAYS));
setlocale(LC_ALL, 'en_US');

try {
    // Replace with a real token!
    $token = "EXAMPLE TOKEN HERE";

    print "logging in\n";
    $sessionId = $client->login(array('apiToken' => $token))->return;

    $session_header = new SoapHeader("http://api.bronto.com/v4",
        'sessionHeader',
        array('sessionId' => $sessionId)
    );
    $client->__setSoapHeaders(array($session_header));

    // Replace with a real deliveryId
    $deliveryIds = array('EXAMPLE DELIVERYID');

    // Replace with a real messageId
    $messageIds = array('EXAMPLE MESSAGEID');

    // Replace with a real messageRuleId
    $messageRuleIds = array('EXAMPLE MESSAGERULEID');

    // Replace with a real deliveryGroupId
    // This will be the delivery group the delivery, message, and
    // automated message rules specified above are added to
    $deliveryGroup = array('id' => 'EXAMPLE DELIVERYGROUPID');

    $write_result = $client->addToDeliveryGroup(array('deliveryGroup' =>
        $deliveryGroup,
                                'deliveryIds' => $deliveryIds,
                                'messageIds' => $messageIds,
                                'messageRuleIds' => $messageRuleIds
        ))->return;
}

```

```

    if ($write_result->errors) {
        print "There was a problem adding item to the delivery group:\n";
        print_r($write_result->results);
    } else {
        print "The items were successfully added to the delivery group. Id:
" . $write_result->results[0]->id . "\n";
    }

} catch (Exception $e) {
    print "uncaught exception\n";
    print_r($e);
}

?>

```

addToList

The `addToList` function allows you to add one or more contacts to a list. You are limited to adding 5k contacts per call.

Syntax

```
writeResult = bApi.addToList(mailListObject list, contactObject[] contacts);
```

Required and Optional List Object Attributes

Name	Type	Required	Description
id	string	Yes (unless the list name is used)	The unique id assigned to the list. You can obtain the id for a list by calling readLists , or by looking at the footer when viewing the overview page for an individual list in the application.
name	string	Yes (unless name the list id is used)	The internal name of the list.

Required and Optional Contact Object Attributes

Name	Type	Required	Description
id	string	Yes (unless name the contact email is used)	The unique id assigned to the contact. You can obtain the id for a contact by calling readContacts , or by looking at the footer when viewing the overview page for an individual contact in the application. You are limited to adding 5k contacts per call.

Name	Type	Required	Description
email	string	Yes (unless name the contact id is used)	The email address assigned to the contact. The email address can be used to reference a specific contact when using the contact functions.

PHP Code Example

```

<?php
/**
 * This script will incrementally add a contact to a list. The contact will
 * not
 * be dropped from any lists that they have already joined. If you want to
 * add a
 * contact to a list and remove the contact from any existing lists, use
 * the
 * addOrUpdateContacts function. To remove a contact from a list without
 * altering membership to other lists, use the removeFromList function.
 *
 * @copyright Copyright (c) 2018 Oracle + Bronto Software (http://
www.bronto.com)
 */

$client = new SoapClient('https://api.bronto.com/v4?wsdl', array('trace' =>
1,
                                'features' => SOAP_SINGLE_ELEMENT_ARRAYS));

try {
    // Add your API token
    $token = "ADD YOUR API TOKEN";

    print "Logging in\n";
    $sessionId = $client->login(array('apiToken' => $token))->return;

    $session_header = new SoapHeader(
        "http://api.bronto.com/v4",
        'sessionHeader',
        array('sessionId' => $sessionId)
    );
    $client->__setSoapHeaders(array($session_header));

    /**
     * $mailListObject is an array containing the list information.
     * You can pass the list id and/or the list name.
     * The list id is the unique id assigned to the list. You can obtain
     * the id for a list by calling readLists, or by looking at the footer
     * when viewing the overview page for an individual list in the
application.
     * The list name is the internal name for your list. We recommend
     * using the list id instead of the list name to speed up your API
calls.
     */

    // Example with both id and name included.
    $mailListObject = array("id" => "LIST ID",
                            "name" => "LIST NAME");

    // Example with only list id.

```

```

// $mailListObject = array("id" => "LIST ID");

// Example with only internal name.
// $mailListObject = array("name" => "LIST NAME");

/**
 * $contactObject is an array containing the contact information.
 * The contact API id is a unique identifier assigned to a contact.
 * You can find the contact API id for a contact by looking at the footer
when
 * viewing the overview page for an individual contact in the application.
 * The contact email address can also be used as an id when adding
contact(s).
 */

// Example using both id and email.
$contactObject = array("id" => "CONTACT ID",
    "email" => "CONTACT EMAIL ADDRESS");

    print "Adding the contact(s) to the list\n";

// Example with id only.
// $contactObject = array("id" => "CONTACT ID");

// Example with email only.
// $contactObject = array("email" => "CONTACT EMAIL ADDRESS");

// Example with multiple contacts.
// $contact1 = array("email" => "FIRST CONTACT EMAIL");
// $contact2 = array("email" => "SECOND CONTACT EMAIL");
// $contact3 = array("email" => "THIRD CONTACT EMAIL");
// $contactObject = array($contact1, $contact2, $contact3);

    $write_result = $client->addToList(array('list' =>
$mailListObject, 'contacts' => $contactObject))->return;

    if ($write_result->errors) {
        print "There was a problem adding the contact(s) to the list:\n";
        print_r($write_result->results);
    } else {
        print "The contact(s) have been added.\n";
    }
} catch (Exception $e)
    print "uncaught exception\n";
    print_r($e);
}
?>

```

addToSMSKeyword

The addToSMSKeyword function allows you to add contacts to an SMS keyword. If you want to remove people from an SMS keyword, use removeFromSMSKeyword.

Syntax

```

writeResult = bApi.addToSMSKeyword(smsKeywordObject[]
    keyword, contactObject[] contacts);

```

Required and Optional SMS Keyword Object Attributes

Name	Type	Required	Description
id	string	No (Required if a name is not passed in)	The unique id for an SMS keyword.
name	string	No (Required if an id is not passed in)	The name assigned to an SMS keyword.

Required and Optional Contact Object Attributes

Name	Type	Required	Description
id	string	No (Required if an email or mobileNumber is not passed in)	The unique id for a contact.
email	string	No (Required if an id or mobileNumber is not passed in)	The email address assigned to a contact.
mobileNumber	string	No (Required if an email or id is not passed in)	The mobile number stored for a contact; must include a valid country code.

SOAP Update Functions

Update functions allow you to make changes to existing data in the Bronto platform. If you want to add data to Bronto you should use an add function.

updateAccounts Multi-Brand Only

The `updateAccounts` function allows you to change account options, settings, and other details.

Overview

For accounts marked as inactive, the only action you can perform is to switch the account to active.

Syntax

```
writeResult = bApi.updateAccounts(accountObject[] accounts);
```

Required and Optional Account Attributes

Name	Type	Required	Description
id	string	Yes	The unique id for the account.
name	string	No	The name for the account. The name can be used to reference a specific account when using the account functions.

Name	Type	Required	Description
status	string	Yes	The status of the account. You can update an account from <code>unrestricted</code> to <code>inactive</code> and <code>inactive</code> to <code>unrestricted</code> . You can't change the status of a <code>restricted</code> account. Users will not be able to login (from the UI or the API) to accounts with a status of <code>inactive</code> .
generalSettings	GeneralSettings[]	Yes	Allows you to specify an array of values corresponding to the general settings in the account. Click in the Name row for more information on the items in the array.
contactInformation	ContactInformation[]	No	Allows you to specify an array of values corresponding to the contact information for the account. Click in the Name row for more information on the items in the array.
formatSettings	FormatSettings[]	No	Allows you to specify an array of values corresponding to the formatting settings in the account. Click in the Name row for more information on the items in the array.
repliesSettings	RepliesSettings[]	No	Allows you to specify an array of values corresponding to the replies settings in the account. Click in the Name row for more information on the items in the array.
allocations	AccountAllocations[]	Yes	Agency Only Allows you to specify an array of values corresponding to the allocations you assign to the account you are adding. Click in the Name row for more information on the items in the array.

PHP Code Example

```

<?php

/**
 * This example will update the details of an Agency client account.
 * Note that it uses only the minimum required account attributes
 * needed to perform an update
 */

$client = new SoapClient('https://api.bronto.com/v4?wsdl', array('trace' =>
    1,
                                'features' => SOAP_SINGLE_ELEMENT_ARRAYS));

try {
    $token = "ADD YOUR TOKEN HERE";

    print "logging in\n";
    $sessionId = $client->login(array('apiToken' => $token))->return;

    $session_header = new SoapHeader("http://api.bronto.com/v4",
        'sessionHeader',
        array('sessionId' => $sessionId));
    $client->__setSoapHeaders(array($session_header));

    //General Settings
    $generalSettings = array(
        'sitename' => 'someagency',
        'agencyTemplateuploadPerm' => 1,
        'defaultTemplates' => 1,
        'enableInboxPreviews' => 0,
        'allowCustomizedBranding' => 1,
        'bounceLimit' => 7,
        'usageAlertEmail' => 'joe@example.com',
        'sendUsageAlerts' => 1,
    );

    $allocations = array(
        'periodFrequency' => 12,
        'bundle' => 'professional',
    );

    $account = array(
        'id' => 'ADD THE CLIENT ACCOUNT ID HERE',
        'status' => 'active',
        'generalSettings' => $generalSettings,
        'allocations' => $allocations,
    );

    $res = $client->updateAccounts(array($account))->return;

    if ($res->errors) {
        print "There was a problem updating the account:\n";
        print_r($res->results);
    } else {
        print "Account has been updated. \n";
    }
} catch (Exception $e) {
    print "uncaught exception\n";
    print_r($e);
}

```


updateApiTokens

The `updateApiTokens` function allows you to change API token options and settings.

Syntax

```
writeResult = bApi.updateApiTokens(apiTokenObject[] apiTokens);
```

Required and Optional API Token Attributes

Name	Type	Required	Description
id	string	Yes	The unique id for the API token. The id can be used to reference a specific API token when using the <code>apiToken</code> functions.
name	string	No	The name assigned to API token. The name can be used to reference a specific API token when using the <code>apiToken</code> functions.
permissions	int	Yes	<p>The permissions assigned to the API token. An API token can have read, write, and send permissions. Each permission is assigned an int value. To assign:</p> <ul style="list-style-type: none"> • Read = 1 • Write = 2 • Read, Write = 3 • Send = 4 • Read, Send = 5 • Send, Write = 6 • Read, Write, Send = 7 <p> Note: If the permissions field is empty, all token permissions will default to 1.</p>
active	boolean	No	Whether or not the API token is active. You can always go back and activate or deactivate API tokens at a later time.
accountId	boolean	No	The account, referenced by ID, that the API token is assigned to.

updateContacts


The `updateContacts` function allows you to update contact(s) and associated data.

Syntax

```
writeResult = bApi.updateContacts(contactObject[] contacts);
```

Required and Optional Contact Object Attributes

Name	Type	Required	Description
id	string	No	The unique id for the contact. Either id or email is required to locate the object. You can obtain the id for a contact by calling readContacts , or by looking at the footer when viewing the overview page for an individual contact in the application. The id, mobileNumber, or email is required to locate the contactObject .
email	string	No	The email address assigned to the contact. The id, mobileNumber, or email is required to locate the contactObject .
mobileNumber	string	No	The mobile number stored for the contact. The id, mobileNumber, or email is required to locate the contactObject . A valid country code must be included when updating a mobile number for a contact.

Name	Type	Required	Description
status	string	No	<p>The status of the contact. The status can be set to transactional, onboarding, unconfirmed, unsub, or bounce.</p> <p>Notes:</p> <ul style="list-style-type: none"> • You cannot change a contact's status directly to active. Rather, you need to switch their status to onboarding so that they can be run through the automated onboarding process. • While you can change a contact's status to bounce, we recommend you do not do this unless you are doing so as part of a bulk import while moving to Bronto from another platform. • You can not update a contact's status if they are on the suppression list. • If a contact has a status of onboarding, you can update their status to bounce, transactional, unconfirmed, or unsub, but not active. <p> CAUTION: Changing a contact's status can have implications on your ability to send email to them. For an explanation of each type of contact status, view the description for status on the contactObject page.</p>

Name	Type	Required	Description
msgPref	string	No	The message preference for the contact. A contact can have a message preference of text or html.
source	string	No	The source or where the contact came from. The source can manual, import, api, webform, or sforcereport (salesforce report).
customSource	string	No	A source you define that states where the contact came from.

Name	Type	Required	Description
<code>listIds</code>	string, array. Use an array for multiple ids	No	<p>The lists (referenced by ID) that the contact belongs to. You obtain <code>listIds</code> by calling the <code>readLists</code> function.</p> <p>The lists you set in this call are absolute, not incremental, to lists the contact may already be on. This means contacts are removed from any list(s) not specified in this call and will only be added to lists you specify in this call. If you pass an empty array for <code>listId</code> or do not pass the <code>listId</code> parameter, list membership will remain unchanged.</p> <p>If you want to</p> <ul style="list-style-type: none"> Incrementally add a contact to a list without affecting their membership on other lists, use the addToList function. Incrementally remove a contact from a list, use the removeFromList function. <p>If you want to use this call to incrementally add the contact to a new list and retain their current list membership, you'll need to</p> <ul style="list-style-type: none"> Call readContacts. Obtain the ids for the lists the contact is currently a member of. Pass in those ids along with the new list ids when calling <code>updateContacts</code>.
<code>fields</code>	contactField[]	No	An array of the fields and corresponding field data associated with the contact.
<code>SMSKeywordIDs</code>	string, array. Use an array for multiple ids	No	An array of the SMS keyword ids you want to subscribe the contact to.

updateContentTags

The `updateContentTags` function allows you to modify one or more content tags.

Syntax

```
writeResult = bApi.updateContentTags(contentTagObject[] contentTags);
```

Required and Optional Content Tag Attributes

Name	Type	Required	Description
id	string	Yes	The unique id assigned to content tag object. You can obtain the id for a list by calling readContentTags .
name	string	No	<p>The name you assigned to the content tag. The name you specify will be used to reference this block of content via the Content Tag. For example, if you name the Content Tag <code>mycontenttag</code>, you would reference this Content Tag in your message by adding <code>%%@mycontenttag%%</code> to your message. Note:</p> <ul style="list-style-type: none"> • The content tag name must be 100 characters or less. • The name cannot be blank or null
value	string	No	<p>The content that will be displayed when the message is sent. Note:</p> <ul style="list-style-type: none"> • The value cannot contain other content tags, field tags, API message tags, or dynamic code. • HTML can be used in the value, however, the HTML will appear unformatted if the content tag is used in a plain text message.

updateDeliveries

The `updateDeliveries` function allows you to update the send time of a delivery, or cancel an upcoming delivery.

Overview

If you want to change the message used in the delivery, or change the delivery targets, you should first cancel the delivery (set the status to `skipped`), and then schedule a new delivery (`addDeliveries`).

Syntax

```
writeResult = bApi.updateDeliveries(deliveryObject[] delivery);
```

Required and Optional Delivery Attributes

Name	Type	Required	Description
id	string	Yes	The unique id for the delivery.
start	dateTime	No	The date the delivery is scheduled to be sent .
status	string	No	The status of the delivery. Valid values: <code>skipped</code> . Set the status to <code>skipped</code> to cancel the delivery.

updateDeliveryGroup

The `updateDeliveryGroup` function allows you to update delivery groups and associated data.

Syntax

```
writeResult = bApi.updateDeliveryGroup(deliveryGroupObject[] deliveryGroup);
```

Required and Optional Delivery Attributes

Name	Type	Required	Description
id	string	No	The unique id for the delivery group.
name	string	No	The name assigned to the delivery group.

updateFields

The `updateFields` function allows you to modify existing fields.

Overview

You should not use fields to store particularly sensitive or private information about your contacts. Information such as credit card numbers, social security numbers, unencrypted passwords, and other similar data should be stored outside of the application in a system specifically designed for handling this type of data.

Any field data you use in a message must be smaller than 1 MB.

Syntax

```
writeResult = bApi.updateFields(fieldObject[] fields);
```

Required and Optional Field Object Attributes

Name	Type	Required	Description
id	string	Yes	The unique id of the field object.
name	string	Yes	The internal name of the field.
label	string	Yes	The external (public facing) name of the field.
visibility	string	No	The visibility selected for the field {public, private}. Public fields are visible to you and can be made visible to your contacts. Private fields are visible only to you.
options	FieldOptionObject[]	No	The possible options that can be set for a field if the field is a pull-down, check box, or radio button.

updateHeaderFooters

The `updateHeaderFooters` function allows you to modify reusable headers and footers that can be included at the top and bottom of messages.

Syntax

```
writeResult = bApi.updateHeaderFooters(headerFooterObject[] headersFooters);
```

Required and Optional HeaderFooter Object Attributes

Name	Type	Required	Description
id	string	Yes	The unique id of the header/footer object.
name	string	Yes	The name assigned to the header/footer.
html	string	Yes	The HTML version of the header.
text	string	Yes	The text version of the header
isHeader	boolean	No	Set to TRUE if the object is a header.

updateLists

The `updateLists` function allows you to modify one or more lists to the account.

Syntax

```
writeResult = bApi.updateLists(mailListObject[] lists);
```

Required and Optional List Object Attributes

Name	Type	Required	Description
id	string	Yes	The unique id of the list object. You can obtain the id for a list by calling readLists , or by looking at the footer when viewing the overview page for an individual list in the application.
name	string	No	The internal name of the list.
label	string	No	The external (customer facing) name of the list.

updateLogins

The `updateLogins` function allows you to change a login's options and permissions.

Syntax

```
writeResult = bApi.updateLogins(loginObject[] logins);
```

Required and Optional Login Object Attributes

Name	Type	Required	Description
username	string	Yes	The username assigned to the login.
password	string	Yes	The password assigned to the login.
contactInformation	ContactInformation []	Yes	An array of the contact information set for the account. Click in the Type column for more information.
permissionAgencyAdmin	boolean	Yes	Gives the login agency administration permissions if you are creating a login for an agency account.
permissionAdmin	boolean	Yes	Gives the login administrative permission if you are creating a login for a client-account or a professional account.
permissionApi	boolean	Yes	Gives the login API permission if you are creating a login for a client-account or a professional account.

Name	Type	Required	Description
permissionUpgrade	boolean	Yes	Gives the login permission to purchase upgrades, such as inbox preview and additional fields. Applicable if you are creating a login for a client-account or professional account.
permissionFatigueOverride	boolean	Yes	Gives the login permission to override any contact frequency caps you have set for them. Applicable if you are creating a login for a client-account or professional account.
permissionMessageCompose	boolean	Yes	Gives the login permission to create messages if you are creating a login for a client-account or a professional account.
permissionMessageDelete	boolean	Yes	Gives the login permission to delete messages if you are creating a login for a client-account or a professional account.
permissionAutomatorCompose	boolean	Yes	Gives the login permission to create automated message rules if you are creating a login for a client-account or a professional account.
permissionListCreateAndSend	boolean	Yes	Gives the login permission to create and send to lists if you are creating a login for a client-account or a professional account.
permissionListCreate	boolean	Yes	Gives the login permission to create, but not send messages to lists if you are creating a login for a client-account or a professional account.
permissionSegmentCreate	boolean	Yes	Gives the login permission to create segments if you are creating a login for a client-account or a professional account.

Name	Type	Required	Description
permissionFieldCreate	boolean	Yes	Gives the login permission to create fields if you are creating a login for a client-account or a professional account.
permissionFieldRecord	boolean	Yes	Gives the login permission to create messages if you are creating a login for a client-account or a professional account.
permissionSubscriberCreate	boolean	Yes	Gives the login permission to create contacts if you are creating a login for a client-account or a professional account.
permissionSubscriberView	boolean	Yes	Gives the login permission to view contacts if you are creating a login for a client-account or a professional account.

updateMessageFolders

The `updateMessageFolders` function allows you to change message folders for your account. Message folders can be used to group messages.

Syntax

```
writeResult = bApi.updateMessageFolders(messageFolderObject[]
    messageFolders);
```

Required and Optional MessageFolder Object Attributes

Name	Type	Required	Description
id	string	Yes	The unique id of the message folder object.
name	string	No	The name assigned to the folder.
parentId	string	No	The unique id assigned to the parent folder which contains this folder.

updateMessageRules

The `updateMessageRules` function allows you to change Automated Message Rules which used to trigger automatic email delivery to groups of contacts.

Syntax

```
writeResult = bApi.updateMessageRules(messageRuleObject[] messageRules);
```


Required and Optional MessageRule Object Attributes

Name	Type	Required	Description
id	string	Yes	The unique id of the message rule object.
name	string	No	The name assigned to the Automated Message Rule.
messageId	string	No	The unique id of the Message that will be sent via the Automated Message Rule.

updateMessages

The `updateMessages` function allows you to change the content of messages to your account.

Overview

Messages created using the email message editor cannot be updated using this call. Currently the API does not support the structure of email message editor messages. You can use `updateMessage` to update HTML or plain text messages, which includes messages created using the WYSIWYG editor.

Syntax

```
writeResult = bApi.updateMessages(messageObject[] messages);
```

Required and Optional Message Object Attributes

Name	Type	Required	Description
id	string	Yes	The unique id of the message object. You can obtain the id for a message by calling readMessages , or by looking at the footer when viewing the overview page for an individual message in the application.
name	string	No	the name of this message
messageFolderId	string	No	the id of the folder containing this message
content	MessageContentObject[]	No	array of contents for this message


updateSMSDeliveries

The `updateSMSDeliveries` function allows you to update SMS deliveries.

Syntax

```
writeResult = bApi.updateSMSDeliveries(smsDeliveryObject[] deliveries);
```

Required and Optional Delivery Object Attributes

Name	Type	Required	Description
id	string	Yes	The unique id assigned to an SMS delivery.
start	dateTime	No	The date the delivery is scheduled to be sent.
messageId	string	No	The id assigned to the SMS message being used in the SMS delivery.
status	string	No	The status of the SMS delivery. Valid values for this function are: <ul style="list-style-type: none"> skipped
content	string	Yes	The content used in the SMS delivery. <div>  Note: SMS messages are limited to 160 characters. The following text must be included somewhere in the body of the SMS message: Text STOP to end </div>
recipients	deliveryRecipientObject[]	Yes	An array of the recipients who are scheduled to receive the SMS delivery.
fields	smsMessageFieldObject[]	No	An array of the API fields and data to substitute into the SMS message being sent by this SMS delivery.

updateSMSKeywords

The `updateSMSKeywords` function allows you to update an SMS keyword.

Overview

If you want to add contacts to an SMS keyword, use `addToSMSKeyword`. If you want to remove contacts from an SMS keyword, use `removeFromSMSKeyword`.

Syntax

```
writeResult = bApi.updateSMSKeywords(smsKeywordObject[] keywords);
```

Required and Optional SMS Keyword Object Attributes

Name	Type	Required	Description
id	string	No (Required if a name is not passed in)	The unique id for the SMS keyword.
name	string	No (Required if an id is not passed in.)	The name assigned to the SMS keyword.
description	string	No	The description provided for the SMS keyword.
subscriberCount	long	No	The number of people subscribed to the SMS keyword.
frequencyCap	long	No	The frequency cap represents the maximum number of SMS messages you can send to a person subscribed to the keyword each month. Best practice suggests you set the frequency cap to 30 or less per month. Valid values are 1-30.
dateCreated	dateTime	No	The date and time the SMS keyword was created.
scheduledDeleteDate	dateTime	No	When you delete an SMS keyword, it is marked for deletion and then deleted 7 days later. The scheduledDeleteDate represents the date the SMS keyword will be deleted.
confirmationMessage	string	No	The confirmation message set for the SMS keyword. The confirmation message is sent when a user confirms their subscription to the SMS keyword. The text “ <i>Txt STOP to <XXXXX> to end, HELP for info. <XX>msg/mo. Msg&Data rate may apply.</i> ” will be appended to your message. The confirmationMessage can be a maximum of 83 characters.

Name	Type	Required	Description
messageContent	string	No	The message content set for the SMS keyword. The messageContent will be sent each time a contact texts into the keyword or replies to a message from the keyword. The messageContent can be a maximum of 160 characters.
keywordType	string	No	<p>The type set for the SMS keyword. Valid values are:</p> <ul style="list-style-type: none"> • basic – Basic keywords are non-subscription keywords meant for individual transactions. With basic keywords, a person texts into a keyword and a response is sent back. The interaction ends there. The recipient is not added to a list because they have not agreed to receive future marketing messages from you. • subscription – Subscription based keywords require a person to choose to receive SMS messages from you by texting into a given keyword. Contacts who subscribe to a subscription based keyword will be added to a list so that you can send SMS messages to them in the future. • text2join – text2join keywords allows you to grow your list by providing a way for potential contacts to text their email address in and automatically be added to one of your lists.

updateSMSMessages

The `updateSMSMessages` function allows you to update SMS messages in your account.

Syntax

```
writeResult = bApi.updateSMSMessages(smsMessageObject[] messages);
```

Required and Optional SMS Message Object Attributes

Name	Type	Required	Description
id	string	Yes	The unique id assigned to an SMS message.
name	string	No	The name of the SMS message.
messageFolderId	string	No; default is the root (top-level) folder	The id of a folder to add the SMS message to.
shortenUrls	boolean	No	Indicates if the SMS message uses shortened URLs. In addition to shortening URLs, which allows for more content, this setting also allows the application to track the URLs used in the message. URLs will be shortened when the message is sent. The character count takes this into account, and thus represents the count for the message as if the URLs were shortened.
content	string	No	The content for the SMS message. SMS messages are limited to 160 characters. The following text must be included somewhere in the body of the SMS message: Text STOP to end

SOAP Delete Functions

Delete functions help you to remove data from the Bronto platform. Always carefully consider what impact deleting data can have on other aspects of the platform.

For example, if you delete a message make sure you update any future deliveries that message may be associated with.

deleteAccounts Multi-Brand Only

The `deleteAccounts` function allows you to remove a subaccount from a Multi-Brand account.

Syntax

```
writeResult = bApi.deleteAccounts(accountObject[] accounts);
```

Required Account Object Attributes

Name	Type	Required	Description
id	string	Yes	The unique id for the account.

deleteApiTokens

The `deleteApiTokens` function allows you to remove an API token.

Syntax

```
writeResult = bApi.deleteApiTokens(apiTokenObject[] apiTokens);
```

Required API Token Object Attributes

Name	Type	Required	Description
id	string	Yes	The unique id for the API token.

deleteContacts

The `deleteContacts` function allows you to remove one or more contacts from your contact database with Bronto.

Syntax

```
writeResult = bApi.deleteContacts(contactObject[] contacts);
```

Required Contact Object Attributes

Name	Type	Required	Description
id	string	Yes	The unique id for the contact. You can obtain the id for a contact by calling readContacts , or by looking at the footer when viewing the overview page for an individual contact in the application.

deleteContentTags

The `deleteContentTags` function allows you to delete one or more content tags.

Syntax

```
writeResult = bApi.deleteContentTags(contentTagObject[] contentTags);
```

Required List Object Attributes

Name	Type	Required	Description
id	string	Yes	The unique id assigned to the content tag.

deleteDeliveries

The `deleteDeliveries` function allows you to delete email deliveries.

Syntax

```
writeResult = bApi.deleteDeliveries(deliveryObject[] delivery);
```

Required Delivery Object Attributes

Name	Type	Required	Description
id	string	Yes	The unique id for an email delivery you want to delete.

deleteDeliveryGroup

The `deleteDeliveryGroup` function allows you to delete a delivery group.

Syntax

```
writeResult = bApi.deleteDeliveryGroup(deliveryGroupObject[] deliveryGroup);
```

Required Delivery Group Object Attributes

Name	Type	Required	Description
id	string	no	The unique id assigned to the delivery group you want to delete. Use only if you want to delete an entire delivery group.

deleteFields

The `deleteFields` function allows you to delete fields and field data.

Syntax

```
writeResult = bApi.deleteFields(fieldObject[] fields);
```

Required And Optional Field Object Attributes

Name	Type	Description
id	string	The unique id of the field.

deleteFromDeliveryGroup

The `deleteFromDeliveryGroup` function allows you to delete deliveries, messages, or automated message rules from a delivery group, without actually deleting the delivery group.

Syntax

```
writeResult = bApi.deleteFromDeliveryGroup(deliveryGroupObject[]
    deliveryGroup);
```

Required Delivery Group Object Attributes

Name	Type	Required	Description
deliveryGroup	string	Yes	The id associated with the delivery group you want to delete items from.
deliveryIds[]	string	No	An array of the delivery ids you want to delete from the delivery group.
messageRuleIds[]	string	No	An array of the automated message rule ids you want to delete from the delivery group.
messageIds[]	string	No	An array of the message ids you want to delete from the delivery group.

PHP Code Example

```
<?php
/**
 * This script will delete a delivery, message, and automated message rule
 * from a delivery group.
 *
 * @copyright Copyright (c) 2011 Bronto Software (http://www.bronto.com)
 */

$client = new SoapClient('https://api.bronto.com/v4?wsdl', array('trace' =>
    1,
    'features' => SOAP_SINGLE_ELEMENT_ARRAYS));
setlocale(LC_ALL, 'en_US');

try {
    // Replace with a real token!
    $token = "EXAMPLE TOKEN HERE";

    print "logging in\n";
    $sessionId = $client->login(array('apiToken' => $token))->return;

    $session_header = new SoapHeader("http://api.bronto.com/v4",
        'sessionHeader',
        array('sessionId' => $sessionId)
    );
    $client->__setSoapHeaders(array($session_header));

    // Replace with a real deliveryId
    $deliveryIds = array('EXAMPLE DELIVERYID');

    // Replace with a real messageId
    $messageIds = array('EXAMPLE MESSAGEID');
```



```

// Replace with a real messageId
$messageRuleIds = array('EXAMPLE MESSAGERULEID');

// Replace with a real deliveryGroupId
// This will be the delivery group the delivery, message, and
// automated message rules specified above are deleted from
$deliveryGroup = array('id' => 'EXAMPLE DELIVERYGROUPID');

$write_result = $client->deleteFromDeliveryGroup(array('deliveryGroup' =>
$deliveryGroup,
                                'deliveryIds' => $deliveryIds,
                                'messageIds' => $messageIds,
                                'messageRuleIds' => $messageRuleIds
                                ))->return;

if ($write_result->errors) {
    print "There was a problem deleting items from the delivery group:
\n";
    print_r($write_result->results);
} else {
    print "The items were successfully deleted from the delivery group.
Id: " . $write_result->results[0]->id . "\n";
}

} catch (Exception $e) {
    print "uncaught exception\n";
    print_r($e);
}

?>

```

deleteHeaderFooters

The deleteHeaderFooters function allows you to delete existing headers and footers.

Syntax

```
writeResult = bApi.deleteHeaderFooters(headerFooterObject[] headersFooters);
```

Required HeaderFooter Object Attributes

Name	Type	Required	Description
id	string	Yes	The unique id of assigned to the header/footer.

deleteLists

The deleteLists function allows you to delete an existing list.

Syntax

```
writeResult = bApi.deleteLists(mailListObject[] lists);
```

Required List Object Attributes

Name	Type	Required	Description
id	string	Yes	The unique id assigned to the list. You can obtain the id for a list by calling readLists , or by looking at the footer when viewing the overview page for an individual list in the application.

deleteLogins

The `deleteLogins` function allows you to delete an existing list.

Syntax

```
writeResult = bApi.deleteLogins(loginObject[] logins);
```

Required Login Object Attributes

Name	Type	Required	Description
username	string	Yes	The username of the login.

deleteMessageFolders

The `deleteMessageFolders` function allows you to remove message folders.

Syntax

```
writeResult = bApi.deleteMessageFolders(messageFolderObject[]
messageFolders);
```

Required MessageFolder Object Attributes

Name	Type	Required	Comments
id	string	Yes	The unique id assigned to the folder.

deleteMessageRules

The `deleteMessageRules` function allows you to remove automated message rules.

Syntax

```
writeResult = bApi.deleteMessageRules(messageRuleObject[] messageRules);
```

Required MessageRule Object Attributes

Name	Type	Required	Description
id	string	Yes	The unique id assigned to the Automated Message Rule.

deleteMessages

The `deleteMessages` function allows you to delete existing messages.

Syntax

```
writeResult = bApi.deleteMessages(messageObject[] messages);
```

Required Message Object Attributes

Name	Type	Required	Description
id	string	Yes	The unique id of assigned to the message. You can obtain the id for a message by calling readMessages , or by looking at the footer when viewing the overview page for an individual message in the application.

deleteOrders

The `deleteOrders` function allows you to delete an order.

Syntax

```
writeResult = bApi.deleteOrders(orderObject[] orders);
```

Required Order Object Attributes

Name	Type	Required	Description
id	string	Yes	The unique id for the order.

deleteSMSDeliveries

The `deleteSMSDeliveries` function allows you to delete SMS deliveries.

Syntax

```
writeResult = bApi.deleteSMSDeliveries(smsDeliveryObject[] deliveries);
```

Required Delivery Object Attributes

Name	Type	Required	Description
id	string	Yes	The unique id associated with an SMS delivery.

deleteSMSKeywords

The `deleteSMSKeywords` function allows you to delete an SMS keyword.

Overview

If you want to add people to an SMS keyword, use `addToSMSKeyword`. If you want to remove people from an SMS keyword, use `removeFromSMSKeyword`.

Syntax

```
writeResult = bApi.deleteSMSKeywords (smsKeywordObject[] keywords);
```

Required And Optional SMS Keyword Object Attributes

Name	Type	Required	Description
id	string	No (Required if an name is not passed in)	The unique id for an SMS keyword.
name	string	No (Required if an id is not passed in)	The name assigned to the SMS keyword.

deleteSMSMessages

The `deleteSMSMessages` function allows you to delete new SMS messages from your account.

Syntax

```
writeResult = bApi.deleteSMSMessages (smsMessageObject[] messages);
```

Required And Optional SMS Message Object Attributes

Name	Type	Required	Description
id	string	Yes	The unique id assigned to an SMS message.

SOAP Misc Functions

Miscellaneous functions are functions that do not easily fit into one of the other categories provided.

clearLists

The `clearLists` function is deprecated.

Overview

The `clearLists` function allows you to remove all contacts from a list, hence leaving it completely empty. This does not delete the contacts from your account. The `clearLists` call:

- does not generate events for contacts removed from the affected lists.
- will not trigger workflows when contacts are removed from the affected lists.
- might result in incorrect membership for segments that reference the affected lists.

If you need any of the above behaviors, you should not use the `clearLists` function. You should call [readContacts](#) using the list as a filter to get a list of contact IDs and then call [removeFromList](#) to remove these contacts from the list.

Syntax

```
writeResult = bApi.clearLists (mailListObject[] lists);
```

Required and Optional List Object Attributes

Name	Type	Required	Description
id	string	Yes (unless the list name is used)	The unique id assigned to the list. You can obtain the id for a list by calling readLists , or by looking at the footer when viewing the overview page for an individual list in the application.
name	string	Yes (unless name the list id is used)	The internal name of the list.

PHP Code Example

```
<?php
/*
This example will clear two lists of all contacts on those lists.
You must edit the code to refer to the list id you wish to clear.
*/
$client = new SoapClient('https://api.bronto.com/v4?wsdl', array(
    'trace' => 1,
    'features' => SOAP_SINGLE_ELEMENT_ARRAYS
));
setlocale(LC_ALL, 'en_US');
try {
    $token = "YOUR API TOKEN";
    print "logging in ";
    $sessionId = $client->login(array(
        'apiToken' => $token
    ))->return;
    $session_header = new SoapHeader("http://api.bronto.com/
v4", 'sessionHeader', array(
        'sessionId' => $sessionId
    ));
    $client->__setSoapHeaders(array(
        $session_header
    ));
    $ids = array(
        array(
            'id' => 'YOUR FIRST LIST ID'
        ),
        array(
            'id' => 'YOUR SECOND LIST ID'
        )
    );
    $res = $client->clearLists($ids)->return;
    if ($res->errors) {
        print "There was a problem clearing your lists: ";
        print $res->results[$res->errors[0]]->errorString . " ";
    }
    else {
        print "Lists have been cleared ";
    }
}

catch(Exception $e) {
    print "uncaught exception\n";
}
```

```

    print_r($e);
}

?>

```

login

The `login` function is used to begin an authenticated API session.

Overview

If the attempt is successful, a `sessionId` will be returned that should be included as a SOAP header element for all subsequent requests in the session. Sessions will automatically be expired after 20 minutes of inactivity, at which point you will need to call `login` again.

Syntax

```
result = bApi.login(apiToken);
```

login Parameters

Name	Type	Required	Description
apiToken	string	Yes	The API token is the unique string that you use to access the API. Tokens are generated inside the Bronto application under Settings -> Data Exchange, or via a call to <code>updateAccounts</code> with an account object with an new API token.

PHP Code Example

```

/**
 * This script will log you in and generate a session.
 *
 * @copyright Copyright (c) 2018 Bronto Software
 */

$client = new SoapClient('https://api.bronto.com/v4?wsdl',
    array('trace' => 1, 'features' =>
SOAP_SINGLE_ELEMENT_ARRAYS));

try {
    // Add your API token
    $token = "ADD YOUR API TOKEN";

    $sessionId = $client->login(array('apiToken' => $token))->return;

    $session_header = new SoapHeader("http://api.bronto.com/v4",
        'sessionHeader',
        array('sessionId' => $sessionId));
    $client->__setSoapHeaders(array($session_header));

    print "logging in with sessionId: $sessionId\n";
}

```

```

    } catch (Exception $e) {
        print "uncaught exception\n";
        print_r($e);
    }

```

Python Code Example

```

"""
This example script will login to the API and obtain
a session id.
@copyright Copyright (c) 2018 Bronto Software
"""

import sys
import logging
from suds.client import Client
from suds import WebFault

# Bronto API WSDL
BRONTO_WSDL = 'https://api.bronto.com/v4?wsdl'

# start up basic logging
logging.basicConfig()

# Replace the placeholder text with a valid
# API token
TOKEN = "ADD YOUR API TOKEN"

# login using the token to obtain a session ID
bApi = Client(BRONTO_WSDL)
# print bApi

try:
    session_id = bApi.service.login(TOKEN)
    print "logging in with sessionId: " + session_id
# Just exit if something goes wrong
except WebFault, e:
    print '\nERROR MESSAGE: '
    print e
    sys.exit()

```

Java Code Example

```

/**
 * This script will log you in and generate a session using the
 * <a href="https://github.com/bronto/bronto-api-java-sdk">Bronto
Java SDK</a>.
 * @copyright Copyright (c) 2018 Bronto Software
 */

import com.bronto.api.*;
import com.bronto.api.model.*;
import static com.bronto.api.model.ObjectBuilder.newObject;

public class Login {

    public static void main(String[] args) {

        String apiService = "YOUR API KEY";
        BrontoApi client = new BrontoClient(apiToken);
    }
}

```

```
String sessionId = client.login();
System.out.println("sessionId = " + sessionId);

}
}
```

logout

The logout function is used to immediately end an authenticated API session.

Overview

When successful the sessionId is no longer valid. You will need to use the login function to establish a new authenticated API session.

Syntax

```
result = bApi.logout();
```

PHP Code Example

```
<?php
/**
 * This script will end your API session
 *
 * @copyright Copyright (c) 2016 Bronto Software (http://www.bronto.com)
 */

$client = new SoapClient('https://api.bronto.com/v4?wsdl', array('trace' =>
1,
'features' => SOAP_SINGLE_ELEMENT_ARRAYS));

try {
    $token = "A VALID API TOKEN HERE";

    print "logging in\n";
    $sessionId = $client->login(array('apiToken' => $token))->return;

    $session_header = new SoapHeader("https://api.bronto.com/v4",
        'sessionHeader',
        array('sessionId' => $sessionId));

    $client->__setSoapHeaders(array($session_header));

    // do some work....

    print "logging out\n";
    $client->logout();
} catch (Exception $e) {
    print "uncaught exception\n";
    print_r($e);
}
```

removeFromList

The removeFromList function allows you to remove one or more contacts from a list. This does not delete the contact from your account.

Syntax

```
writeResult = bApi.removeFromList(mailListObject lists, responseObject[]
contacts);
```

Required and Optional List Object Attributes

Name	Type	Required	Description
id	string	Yes (unless the list name is used)	The unique id assigned to the list. You can obtain the id for a list by calling readLists , or by looking at the footer when viewing the overview page for an individual list in the application.
name	string	Yes (unless name the list id is used)	The internal name of the list.

Required and Optional Contact Object Attributes

Name	Type	Required	Description
id	string	Yes (unless name the contact email is used)	The unique id assigned to the contact. You can obtain the id for a contact by calling readContacts , or by looking at the footer when viewing the overview page for an individual contact in the application.
email	string	Yes (unless name the contact id is used)	The email address assigned to the contact. The email address can be used to reference a specific contact when using the contact functions.

PHP Code Example

```
<?php
/**
 * This script will incrementally remove a contact from a list. The
 * contact will only
 * be removed from one list.
 *
 * @copyright Copyright (c) 2018 Oracle + Bronto Software (http://
www.bronto.com)
 */

$client = new SoapClient('https://api.bronto.com/v4?
wsdl', array('trace' => 1,
'features' =>
SOAP_SINGLE_ELEMENT_ARRAYS));
try {
```

```

// Add your API token
$token = "ADD YOUR API TOKEN";

print "Logging in\n";
$sessionId = $client->login(array('apiToken' => $token))->return;

$session_header = new SoapHeader(
    "http://api.bronto.com/v4",
    'sessionHeader',
    array('sessionId' => $sessionId)
);
$client->__setSoapHeaders(array($session_header));

/**
 * $mailListObject is an array containing the list information.
 * You can pass the list id and/or the list name.
 * The list id is the unique id assigned to the list. You can
obtain
 * the id for a list by calling readLists, or by looking at the
footer
 * when viewing the overview page for an individual list in the
application.
 * The list name is the internal name for your list. We recommend
 * using the list id instead of the list name to speed up your API
calls.
 */

// Example with both id and name included.
$mailListObject = array("id" => "LIST ID",
    "name" => "LIST NAME");

// Example with only list id.
// $mailListObject = array("id" => "LIST ID");

// Example with only internal name.
// $mailListObject = array("name" => "LIST NAME");

/** $contactObject is an array containing the contact information.
 * The contact API id is a unique identifier assigned to a contact.
 * You can find the contact API id for a contact by looking at the
footer when
 * viewing the overview page for an individual contact in the
application.
 * The contact email address can also be used as an id when adding
contact(s).
 */

// Example using both id and email.
$contactObject = array("id" => "CONTACT ID",
    "email" => "CONTACT EMAIL ADDRESS");

// Example with id only.
// $contactObject = array("id" => "CONTACT ID");

// Example with email only.
// $contactObject = array("email" => "CONTACT EMAIL ADDRESS");

// Example with multiple contacts.
// $contact1 = array("email" => "FIRST CONTACT EMAIL");
// $contact2 = array("email" => "SECOND CONTACT EMAIL");
// $contact3 = array("email" => "THIRD CONTACT EMAIL");
// $contactObject = array($contact1, $contact2, $contact3);

```

```

        $write_result = $client->removeFromList(array('list' =>
$mailListObject, 'contacts' => $contactObject))->return;

        if ($write_result->results[0]->isError) {
            print "There was a problem removing the contact from the list.
\n";
        } else {
            print "The contact has been removed. Id: " . $write_result-
>results[0]->id . "\n";
        }
    } catch (Exception $e) {
        print "uncaught exception\n";
        print_r($e);
    }
}
?>

```

removeFromSMSKeyword

The `removeFromSMSKeyword` function allows you to remove contacts from an SMS keyword. If you want to add contacts to an SMS keyword, use `addToSMSKeyword`.

Syntax

```

writeResult = bApi.removeFromSMSKeyword(smsKeywordObject[]
keyword, contactObject[] contacts);

```

Required and Optional SMS Keyword Object Attributes

Name	Type	Required	Description
id	string	Yes	The unique id for an SMS keyword.

Required and Optional Contact Object Attributes

Name	Type	Required	Description
id	string	Yes	The unique id for a contact.

PHP Code Example

```

<?php
/**
 * This script will remove a contact from a SMS keyword. The contact
will only
 * be dropped from the specified keyword. You can remove multiple
contacts at a
 * time.
 * @copyright Copyright (c) 2018 Oracle + Bronto Software (http://
www.bronto.com)
 */

$client = new SoapClient('https://api.bronto.com/v4?wsdl', array('trace'
=> 1,
                                'features' =>
SOAP_SINGLE_ELEMENT_ARRAYS));
try {

```

```

// Add your API token
$token = "ADD YOUR API TOKEN";

print "Logging in\n";
$sessionId = $client->login(array('apiToken' => $token))->return;

$session_header = new SoapHeader(
    "http://api.bronto.com/v4",
    'sessionHeader',
    array('sessionId' => $sessionId)
);
$client->__setSoapHeaders(array($session_header));

/** $smsKeywordObject is an array containing SMS keyword information
 * including the id of the keyword and the subscriber count.
 * You can find the SMS keyword API id for a keyword by looking at
the footer
 * when viewing the overview page for a keyword in the application.
 */

// Example with both id and name included.
$smsKeywordObject = array("id" => "SMS KEYWORD ID");

/** $contactObject is an array containing the contact information.
 * The contact API id is a unique identifier assigned to a contact.
 * You can find the contact API id for a contact by looking at the
footer when
 * viewing the overview page for an individual contact in the
application.
 * The contact email address can also be used as an id when adding
contact(s).
 */

// Example using both id and email.
$contactObject = array("id" => "CONTACT ID");

// Example with multiple contacts.
// $contact1 = array("id" => "FIRST CONTACT ID");
// $contact2 = array("id" => "SECOND CONTACT ID");
// $contactObject = array($contact1, $contact2);

$write_result = $client->removeFromSMSKeyword(array('keyword' =>
$smsKeywordObject, 'contacts' => $contactObject))->return;

if ($write_result->results[0]->isError) {
    print "There was a problem removing the contact from the
keyword:\n";
    print_r($write_result->results);
} else {
    print "The contact has been removed from the keyword. Id: " .
$write_result->results[0]->id . "\n";
}
} catch (Exception $e) {
    print "uncaught exception\n";
    print_r($e);
}

```

SOAP Read Functions

Read functions allow you to retrieve information. Often you may need to use a read function to get data that you will need to use to complete an API task.

readAccounts MultiBrand Only

The `readAccounts` function attempts to return the requested data for accounts that match the given `accountFilter`.

Results

The `readAccounts` function may return 1 or many account objects. See the documentation on the `accountObject` for a list of the data fields that could potentially be returned.



Note: The data fields returned depend on the type of data you ask for in your request.

Syntax

```
accountObject[] accounts = bApi.readAccounts( filter accountFilter,
                                             includeinfo, status, pageNumber );
```

readAccounts Parameters

Name	Type	Required	Description
<code>filter</code>	accountFilter	Yes	The <code>accountFilter</code> allows you to read data from a specific account(s).
<code>includeInfo</code>	boolean	Yes	Returns the GeneralSettings[] , ContactInformation[] , FormatSettings[] , RepliesSettings[] , and AccountAllocations[] associated with the account.
<code>status</code>	string	No	The status of the account. Valid values are <code>unrestricted</code> , <code>restricted</code> , and <code>inactive</code> . <code>unrestricted</code> means the account is active and can function normally. <code>inactive</code> means the account has been manually deactivated and cannot send mail. Users will not be able to login (from the UI or the API) to accounts with a status of <code>inactive</code> . <code>restricted</code> means the account has a low sender rating and its outgoing messages are being throttled. <code>restricted</code> and <code>inactive</code> accounts can't be updated using updateAccounts .

Name	Type	Required	Description
pageNumber	int	Yes	Retrieves the next “batch” of objects as the value specified increases from 1. In order to obtain an entire set of objects for a given call, you should increase the number value assigned to pageNumber until no more objects are returned.

PHP Code Example

```

<?php
/**
 * This example will read the details of your Bronto account and
 * print out both the name of the account, and the current number of
 * active contacts in the account. You can filter for a specific
 * accountObject
 * by account name.
 */

$client = new SoapClient('https://api.bronto.com/v4?wsdl', array('trace' =>
1,
                        'features' => SOAP_SINGLE_ELEMENT_ARRAYS));

try {
    $token = "YOUR API TOKEN";

    print "logging in\n";
    $sessionId = $client->login(array('apiToken' => $token))->return;

    $session_header = new SoapHeader("http://api.bronto.com/v4",
                                    'sessionHeader',
                                    array('sessionId' => $sessionId));
    $client->__setSoapHeaders(array($session_header));

    // example with account name filter
    $filter = array('type' => 'AND',
                    'name' => array('operator' => 'EqualTo',
                                    'value' => 'YOUR ACCOUNT NAME TO FILTER')
                    );

    // example with no filter
    //$filter = array();

    print "reading account details\n";
    $accounts = $client->readAccounts(array('pageNumber' => 1,
                                        'includeInfo' => true,
                                        'filter' => $filter,
                                        )
    )->return;

    // print matching results
    print_r($accounts);
    foreach ($accounts as $account) {
        print "name: " . $account->name . "\n";
        print "current active contacts: " . $account->generalSettings->
currentContacts . "\n";
    }
}

```

```

    }

    } catch (Exception $e) {
        print "uncaught exception\n";
        print_r($e);
    }

```

readApiTokens

The `readApiTokens` function attempts to return the requested data for accounts that match the given `accountFilter`.

Results

The `readApiTokens` function may return 1 or many API token objects. See the documentation on the `apiTokenObject` for a list of the data fields that could potentially be returned.



Note: The data fields returned depend on the type of data you ask for in your request.

Syntax

```

apiTokenObject[] apiTokens = bApi.readApiTokens( filter apiTokenFilter,
    pageNumber );

```

readAccounts Parameters

Name	Type	Required	Description
filter	apiTokenFilter	Yes	The <code>apiTokenFilter</code> allows you to read data from a specific <code>apiToken(s)</code> .
pageNumber	int	Yes	Retrieves the next “batch” of objects as the value specified increases from 1. In order to obtain an entire set of objects for a given call, you should increase the number value assigned to <code>pageNumber</code> until no more objects are returned.

PHP Code Example

```

<?php
/**
 * This example will read the details of an API token
 */

$client = new SoapClient('https://api.bronto.com/v4?wsdl',
    array('trace' => 1,
        'features' => SOAP_SINGLE_ELEMENT_ARRAYS)
    );

try {
    // Add your API token
    $token = "API TOKEN HERE";

    print "logging in\n";
    $sessionId = $client->login(array('apiToken' => $token))->return;

```

```

$session_header = new SoapHeader("http://api.bronto.com/v4",
    'sessionHeader',
    array('sessionId' => $sessionId)
);
$client->__setSoapHeaders(array($session_header));

// Add in a valid account ID
$filter = array("accountId" => "ADD ACCOUNT ID",);

$api_tokens = $client->readApiTokens(array("filter" => $filter,
    "pageNumber" => 1
    )
    )->return;

// Print matching token names, ids, and permissions
foreach ($api_tokens as $api_token) {
    print "Name: " . $api_token->name . "; ID: " . $api_token->id . ";
    Permissions: " . $api_token->permissions . "\n";
}
} catch (Exception $e) {
    print "uncaught exception\n";
    print_r($e);
}
>>

```

readBounces

The `readBounces` function attempts to return bounce data that matches the passed in filter.

Results

The `readBounces` function may return 1 or many bounce objects. See the documentation on the `bounceObject` for a list of the data fields that could potentially be returned.

Syntax

```
bounceObject[] bounces = bApi.readBounces(filter bounceFilter, pageNumber);
```

readBounces Parameters

Name	Type	Required	Comments
filter	bounceFilter	Yes	The filter used to return specific bounce data.
pageNumber	int	Yes	Retrieves the next “batch” of objects as the value specified increases from 1. In order to obtain an entire set of objects for a given call, you should increase the number value assigned to <code>pageNumber</code> until no more objects are returned.

readContacts

The `readContacts` function attempts to return contacts that match all of the given filters. The specified attributes of the contacts are returned for each matching contact.

Results

The `readContacts` function may return 1 or many contact objects. See the documentation on the [contactObject](#) for a list of the data fields that could potentially be returned. For more information on result limits and paging, see [How To Read Objects](#).



Note: The data fields returned depend on the type of data you ask for in your request.

Syntax

```
contactObject[] contacts = bApi.readContacts( filter contactFilter,
    includeLists, fields, pageNumber, includeSMSKeywords,
    includeGeoIPData,
    includeTechnologyData, includeRFMDData);
```

readContacts Parameters

Name	Type	Required	Description
filter	contactFilter	Yes	The filter used to return a specific contact.
includeLists	boolean	No	The lists, referenced by id, that the contact belongs to.
fields	string, array	No	The contact field(s) that you want returned. Fields are referenced by their unique id. You can pass in a single id as a string, or multiple ids in an array.
pageNumber	int	Yes	Retrieves the next “batch” of objects as the value specified increases from 1. In order to obtain an entire set of objects for a given call, you should increase the number value assigned to pageNumber until no more objects are returned.
includeSMSKeywords	boolean	No	The SMS keywords the contact is subscribed to.
includeGeoIPdata	boolean	No	Includes the following data in the readOnlyContactData object: <ul style="list-style-type: none"> • geoIPCity • geoIPStateRegion • geoIPZip • geoIPCountry • geoIPCountryCode

Name	Type	Required	Description
includeTechnologyData	boolean	No	Includes the following data in the readOnlyContactData object: <ul style="list-style-type: none"> primaryBrowser mobileBrowser primaryEmailClient mobileEmailClient operatingSystem
includeRFMDData	boolean	No	Includes the following data in the readOnlyContactData object: <ul style="list-style-type: none"> firstOrderDate lastOrderDate lastOrderTotal totalOrders totalRevenue averageOrderValue
includeEngagementData	boolean	No	Includes the following data in the readOnlyContactData object: <ul style="list-style-type: none"> lastDeliveryDate lastOpenDate lastClickDate
customSource	string	No	A user-defined source used to indicate where the contact came from.

PHP Code Example

```
<?php /** * This example will log into your account and try to read the
 * contacts 'john.doe@example.com' and 'jane.doe@example.com' * in two
 ways: first, by specifically matching on those email * addresses, and
 then second by matching on any contact whose * email address contains
 'doe@example.com'. It also prints the * matched contacts' status. */
$client = new SoapClient('https://api.bronto.com/v4?wsdl', array('trace' =>
1,
                                'features' => SOAP_SINGLE_ELEMENT_ARRAYS));

try {
    $token = "YOUR_TOKEN_HERE";

    print "logging in\n";
    $sessionId = $client->login(array('apiToken' => $token))->return;

    $session_header = new SoapHeader("http://api.bronto.com/v4",
                                    'sessionHeader',
                                    array('sessionId' => $sessionId));
```

```

$client->__setSoapHeaders(array($session_header));

// set up a filter to read contacts and match on either of two email
addresses
$filter = array('type' => 'OR',
    'email' => array(array('operator' => 'EqualTo',
        'value' => 'john.doe@example.com'
    ),
        array('operator' => 'EqualTo',
            'value' => 'jane.doe@example.com'
        )
    ),
);

print "reading contacts with equalto filter\n";
$contacts = $client->readContacts(array('pageNumber' => 1,
    'includeLists' => false,
    'filter' => $filter,
))->return;

// print matching contact email addresses
foreach ($contacts as $contact) {
    print $contact->email . ': ' . $contact->status . "\n";
}

// set up a filter to read contacts and match on either of two email
addresses
$filter = array('email' => array(array('operator' => 'Contains',
    'value' => 'doe@example.com'
    ),
    ),
);

print "reading contacts with contains filter\n";
$contacts = $client->readContacts(array('pageNumber' => 1,
    'includeLists' => false,
    'filter' => $filter,
))->return;

// print matching contact email addresses
foreach ($contacts as $contact) {
    print $contact->email . ': ' . $contact->status . "\n";
}

} catch (Exception $e) {
    print "uncaught exception\n";
    print_r($e);
}

```

readContactsWithLatestUnsubscribeDate

The `readContactsWithLatestUnsubscribeDate` function attempts to return contacts that match all of the given filters. If the contact has unsubscribed, then `lastUnsubscribeDate` is also returned.

Results

The `readContactsWithLatestUnsubscribeDate` function may return 1 or many contact objects. See the documentation on the [contactObject](#) for a list of the data fields that could potentially be returned. For more information on result limits and paging, see [How To Read Objects](#).



Note: The data fields returned depend on the type of data you ask for in your request.

Syntax

```
contactObject[] contacts = bApi.readContacts( filter contactFilter,
    includeLists, fields, pageNumber, includeSMSKeywords,
    includeGeoIPData,
    includeTechnologyData, includeRFMDData);
```

readContactsWithLatestUnsubscribeDate Parameters

Name	Type	Required	Description
filter	contactFilter	Yes	The filter used to return a specific contact.
includeLists	boolean	No	The lists, referenced by id, that the contact belongs to.
fields	string, array	No	The fields you want returned that the contact belongs to. Fields are referenced by their unique id. You can pass in a single id as a string, or multiple ids in an array.
pageNumber	int	Yes	Retrieves the next “batch” of objects as the value specified increases from 1. In order to obtain an entire set of objects for a given call, you should increase the number value assigned to pageNumber until no more objects are returned.
includeSMSKeywords	boolean	No	The SMS keywords the contact is subscribed to.
includeGeoIPdata	boolean	No	Includes the following data in the readOnlyContactData object: <ul style="list-style-type: none"> • geoIPCity • geoIPStateRegion • geoIPZip • geoIPCountry • geoIPCountryCode

Name	Type	Required	Description
includeTechnologyData	boolean	No	Includes the following data in the readOnlyContactData object: <ul style="list-style-type: none"> primaryBrowser mobileBrowser primaryEmailClient mobileEmailClient operatingSystem
includeRFMDData	boolean	No	Includes the following data in the readOnlyContactData object: <ul style="list-style-type: none"> firstOrderDate lastOrderDate lastOrderTotal totalOrders totalRevenue averageOrderValue
includeEngagementData	boolean	No	Includes the following data in the readOnlyContactData object: <ul style="list-style-type: none"> lastDeliveryDate lastOpenDate lastClickDate
customSource	string	No	A user-defined source used to indicate where the contact came from.

Example Response

```
stdClass Object
(
  [id] => be4214ab43d9-b782-38b47d23033b
  [email] => test@example.com
  [status] => unsub

  [msgPref] => html
  [source] => import
  [customSource] => birthday campaign
  [created] => 2016-11-29T09:36:24-05:00
  [modified] => 2017-10-05T16:58:36-04:00
  [deleted] =>
  [numSends] => 0
  [numBounces] => 0
  [numOpens] => 0
  [numClicks] => 0
  [numConversions] => 0
  [conversionAmount] => 0
)
```

```
[lastUnsubscribeDate] => 2016-11-29T16:44:07Z
)
```

PHP Code Example

```
<?php /** * This example will log into your account and try to read the
 * contacts 'john.doe@example.com' and 'jane.doe@example.com' * in two
 ways: first, by specifically matching on those email * addresses, and
 then second by matching on any contact whose * email address contains
 'doe@example.com'. It also prints the * matched contacts' status. */
$client = new SoapClient('https://api.bronto.com/v4?wsdl', array('trace' =>
1,
                        'features' => SOAP_SINGLE_ELEMENT_ARRAYS));

try {
    $token = "YOUR_TOKEN_HERE";

    print "logging in\n";
    $sessionId = $client->login(array('apiToken' => $token))->return;

    $session_header = new SoapHeader("http://api.bronto.com/v4",
        'sessionHeader',
        array('sessionId' => $sessionId));
    $client->__setSoapHeaders(array($session_header));

    // set up a filter to read contacts and match on either of two email
    addresses
    $filter = array('type' => 'OR',
        'email' => array(array('operator' => 'EqualTo',
            'value' => 'john.doe@example.com'
        ),
            array('operator' => 'EqualTo',
                'value' => 'jane.doe@example.com'
            )
        ),
    );

    print "reading contacts with equalto filter\n";
    $contacts = $client-
>readContactsWithLatestUnsubscribeDate(array('pageNumber' => 1,
        'includeLists' => false,
        'filter' => $filter,
    )
    )->return;

    // print matching contact email addresses
    foreach ($contacts as $contact) {
        print $contact->email . ': ' . $contact->status . "\n";
    }

    // set up a filter to read contacts and match on either of two email
    addresses
    $filter = array('email' => array(array('operator' => 'Contains',
        'value' => 'doe@example.com'
    ),
    ),
    );

    print "reading contacts with contains filter\n";
    $contacts = $client->readContacts(array('pageNumber' => 1,
```

```

        'includeLists' => false,
        'filter' => $filter,
    )
    )->return;

    // print matching contact email addresses
    foreach ($contacts as $contact) {
        print $contact->email . ': ' . $contact->status . "\n";
    }
} catch (Exception $e) {
    print "uncaught exception\n";
    print_r($e);
}

```

readContentTags

The `readContentTags` function attempts to return content tags that match all parameters of the given filter. The specific attributes of the content tags are returned for each matching content tag.

Results

The `readContentTags` function may return 1 or many content tag objects. See the documentation on the `contentTagObject` to view the data that could potentially be returned.

Syntax

```

contentTagObject[] contentTags = bApi.readContentTags(filter
    contentTagFilter,
    pageNumber);

```

readLists Parameters

Name	Type	Required	Comments
filter	contentTagFilter	Yes	The filter used to return specific content tags.
includeContent	boolean	Yes	Set to true if you want to return the content specified for the content tag in it's valueproperty
pageNumber	int	Yes	Retrieves the next “batch” of objects as the value specified increases from 1. In order to obtain an entire set of objects for a given call, you should increase the number value assigned to pageNumber until no more objects are returned.

PHP Code Example

```

<?php
/**
 * This example script will return data for content tags whose name
 * matches the filter

```

```

*/

$client = new SoapClient('https://api.bronto.com/v4?wsdl', array('trace' =>
1,
'features'
=> SOAP_SINGLE_ELEMENT_ARRAYS));

try {
    $token = "API TOKEN HERE";

    print "logging in\n";
    $sessionId = $client->login(array('apiToken' => $token))->return;

    $session_header = new SoapHeader("http://api.bronto.com/v4",
        'sessionHeader',
        array('sessionId' => $sessionId));
    $client->__setSoapHeaders(array($session_header));

    // set up a filter to read content tags
    $filter = array('type' => 'AND',
        'name' => array('operator' => 'Contains',
            'value' => 'SOME EXAMPLE TEXT')
    );

    print "reading all content tags\n";
    $tags = $client->readContentTags(array('pageNumber' => 1,
        'filter' => $filter,
        'includeContent' => true))-
>return;

    // print matching tag id, name, and value.
    foreach ($tags as $tag) {
        print "ID: " . $tag->id . "\n";
        print "Name: " . $tag->name . "\n";
        print "Value: " . $tag->value . "\n";
    }
} catch (Exception $e) {
    print "uncaught exception\n";
    print_r($e);
}

```

Python Code Example

```

import sys
import logging
from suds.client import Client
from suds import WebFault

# This example script will login in to the API and read data
# for a content tag

# BE SURE TO REPLACE ALL PLACEHOLDER TEXT

# Tested with Python 2.7.1 and suds soap library version 0.4

# See suds home page:
# https://fedorahosted.org/suds/

```



```

# Bronto API WSDL
BRONTO_WSDL = 'https://api.bronto.com/v4?wsdl'

# Start up basic logging
logging.basicConfig()

# Replace the placeholder text with a valid
# API token
TOKEN = "ADD API TOKEN HERE"

# Login using the token to obtain a session ID
bApi = Client( BRONTO_WSDL )

try:
    session_id = bApi.service.login(TOKEN)
# Just exit if something goes wrong
except WebFault, e:
    print '\nERROR MESSAGE: '
    print e
    sys.exit()

# Set up the soap headers using the
# session_id obtained from login()
session_header = bApi.factory.create("sessionHeader")
session_header.sessionId = session_id
bApi.set_options(soapheaders=session_header)

# Create the contentTagsFilter passed into
# readContentTags()

filter = bApi.factory.create('contentTagFilter')

stringValue = bApi.factory.create('stringValue')
stringValue.value = 'example'
filterOperator = bApi.factory.create('filterOperator')
stringValue.operator = filterOperator.Contains

filterType = bApi.factory.create('filterType')

# To search by ID, remove the code on line 62, and
# comment out the line below:
# filter.id = 'THE ID OF THE CONTENT TAG'
filter.name = stringValue
filter.type = filterType.AND

pageNumber = 1
includeContent = True

try:
    contentTag = bApi.service.readContentTags(filter, includeContent,
    pageNumber)
except WebFault, e:
    print '\nERROR MESSAGE: '
    print e
    sys.exit()
print bApi.last_sent()
print bApi.last_received()
print contentTag
for tag in contentTag:
    print 'Content Tag ID: ' + tag.id
    print 'Content Tag Name: ' + tag.name

```

```
print 'Content Tag Value: ' + tag.value
```

readConversions

The `readConversions` function attempts to return conversions that match the given filter. You can return the conversions recorded for individual contacts or deliveries depending on your needs.

Results

The `readConversions` function may return 1 or many conversion objects. See the documentation on the `conversionObject` for a list of the data fields that could potentially be returned.



Note: The `readConversions` function is limited to returning 100 conversion objects per page. For more information on result limits and paging, see [How To Read Objects](#).



Note: The data fields returned depend on the type of data you ask for in your request.

Syntax

```
conversionObject[] conversions = bApi.readConversions( filter
    conversionFilter, pageNumber);
```

readConversions Parameters

Name	Type	Required	Description
filter	conversionFilter	Yes	The filter used to return specific conversions. You can specify one filter key per request. For example, you can perform a request for two <code>contactIds</code> , in which case, conversions for both <code>contactIds</code> should be returned. You can not specify both a <code>contactId</code> and <code>deliveryId</code> .
pageNumber	int	Yes	Retrieves the next “batch” of objects as the value specified increases from 1. In order to obtain an entire set of objects for a given call, you should increase the number value assigned to <code>pageNumber</code> until no more objects are returned. <div> Note: The <code>readConversions</code> function is limited to returning 100 conversion objects per page. </div>

PHP Code Example

```
<?php
/**
 * This example will read conversion data for a specific contact
 */

$client = new SoapClient('https://api.bronto.com/v4?wsdl',
    array('trace' => 1,
        'features' => SOAP_SINGLE_ELEMENT_ARRAYS)
    );

try {

    // Add your API token
    $token = "API TOKEN HERE";

    print "logging in\n";
    $sessionId = $client->login(array('apiToken' => $token))->return;

    $session_header = new SoapHeader("http://api.bronto.com/v4",
        'sessionHeader',
        array('sessionId' => $sessionId));
    $client->__setSoapHeaders(array($session_header));

    $filter = array(
        // Add the ID of the contact you want conversion data for
        'contactId' => 'CONTACT ID HERE'
    );

    $conversions = $client->readConversions(array(
        'filter' => $filter,
        'pageNumber' => 1))->return;

    print_r($conversions);

} catch (Exception $e) {
    print "uncaught exception\n";
    print_r($e);
}
?>
```

readDeliveries

The `readDeliveries` function attempts to return deliveries that match all of the given filter. A delivery is a message that has been sent to a specific set of recipients. Although a delivery is associated with a message, the message's content may have changed since the delivery was performed and so should not be used for determining the actual content sent. Instead, the content that was delivered may be included in the return of this function.

Results

The `readDeliveries` function may return 1 or many delivery objects. See the documentation on the `deliveryObject` for a list of the data fields that could potentially be returned. For more information on result limits and paging, see [How To Read Objects](#).



Note: The data fields returned depend on the type of data you ask for in your request.

Syntax

```
deliveryObject[] deliveries = bApi.readDeliveries( filter deliveryFilter
    includeRecipients, includeContent, pageNumber, includeOrderIds);
```

readDeliveries Parameters

Name	Type	Required	Description
filter	deliveryFilter	Yes	A filter used for returning specific deliveries.
includeRecipients	boolean	Yes	Include the recipients of returned deliveries. You will get back an array of the recipients who were, or are scheduled to receive the delivery. If the delivery was sent to a list, you will get back a list ID. If it was sent to a segment, you will get back a segment ID. If it was sent to an individual contact, you will get back a contact ID. If the delivery was sent to a combination of items (lists, segments, and contacts), you will get back an id for each item.
includeContent	boolean	Yes	Include the content of returned deliveries.
pageNumber	int	Yes	Retrieves the next “batch” of objects as the value specified increases from 1. In order to obtain an entire set of objects for a given call, you should increase the number value assigned to pageNumber until no more objects are returned.
includeOrderIds	boolean	No	Include the order IDs associated with returned deliveries. You will get back an array of IDs for orders associated with the delivery. This only returns orderIDs that triggered the delivery; it does not return the Order ID/Cart ID associated with a delivery from a conversion.

PHP Code Example

```

<?php
/**
 * This example will read the details of any delivery made in your Bronto
 * account in the last 24 hours, and print out what message was sent, to how
 * many contacts, how many messages were actually delivered, and how many of
 * those contacts opened or clicked on links in your message.
 */

$client = new SoapClient('https://api.bronto.com/v4?wsdl', array('trace' =>
    1,
                                'features' => SOAP_SINGLE_ELEMENT_ARRAYS));
setlocale(LC_ALL, 'en_US');

try {
    $token = "YOUR_TOKEN_HERE";

    print "logging in\n";
    $sessionId = $client->login(array('apiToken' => $token))->return;

    $session_header = new SoapHeader("http://api.bronto.com/v4",
        'sessionHeader',
        array('sessionId' => $sessionId));
    $client->__setSoapHeaders(array($session_header));

    // compute date/time 24 hours ago
    $startDate = date('c', time() - (2* 24 * 60 * 60)); // 24 hours * 60
    minutes * 60 seconds;

    // set up a filter to read deliveries in the last 24 hours
    $filter = array('start' => array('operator' => 'After',
        'value' => $startDate,
        ),
        'status' => 'sent',
    );

    print "reading deliveries completed from past 24 hours\n";
    $deliveries = $client->readDeliveries(array('pageNumber' => 1,
        'includeRecipients' => false,
        'includeContent' => false,
        'filter' => $filter,
        )
    )->return;

    // print matching results
    foreach ($deliveries as $delivery) {
        // get name of the message sent.
        $msgFilter = array('id' => $delivery->messageId);
        $message = array_pop($client->readMessages(array('pageNumber' => 1,
            'includeContent' => false,
            'filter' => $msgFilter))->return);
        $startString = strftime('%c', strtotime($delivery->start));
        print "Message: \"" . $message->name . "\" sent at: " .
        $startString . "\n";
        print "\tSent: " . $delivery->numSends . "\n\tDelivered: " . $delivery-
        >numDeliveries . " (" .
            number_format(($delivery->numDeliveries / $delivery->numSends) *
            100), 0, '.', ',') . "%)\n";
        print "\tOpens: " . $delivery->numOpens . "\n\tClicks: " . $delivery-
        >numClicks . "\n";
    }
}

```

```

} catch (Exception $e) {
    print "uncaught exception\n";
    print_r($e);
}

```

Python Code Example

```

from suds.client import Client
from suds import WebFault
import sys
import logging

# Tested with Python 2.6.1 and suds soap library version 0.4

# See suds home page:
# https://fedorahosted.org/suds/

# Bronto API WSDL
BRONTO_WSDL = 'https://api.bronto.com/v4?wsdl'

# start up basic logging
logging.basicConfig()

TOKEN = "ADD TOKEN HERE"

# login using the token to obtain a session ID
bApi = Client( BRONTO_WSDL )

try:
    session_id = bApi.service.login(TOKEN)
    # just exit if something goes wrong
except WebFault, e:
    print '\nERROR MESSAGE: '
    print e
    sys.exit()

# Set up the soap headers using the
# session_id obtained from login()
session_header = bApi.factory.create("sessionHeader")
session_header.sessionId = session_id
bApi.set_options(soapheaders=session_header)

# Set up a deliveryFilter that will return deliveries
# made in the specified date range
filter = bApi.factory.create('deliveryFilter')
start = bApi.factory.create('dateValue')
start.operator = "After"
start.value = "2013-05-09T19:20:30-05:00"

end = bApi.factory.create('dateValue')
end.operator = "Before"
end.value = "2013-06-12T19:20:30-05:00"

filter.type = 'AND'
filter.start = [start, end]

```

```

pageNumber = 1

try:
    read_deliveries = bApi.service.readDeliveries(filter, includeRecipients
    = True, includeContent = False, pageNumber = 1)
except WebFault, e:
    print '\nERROR MESSAGE: '
    print e
    sys.exit()
print read_deliveries

```

readDeliveryGroups

The readDeliveryGroups function attempts to return a delivery group and all the items contained in the delivery group.

Results

The readDeliveryGroups function may return 1 or many delivery group objects. See the documentation on the deliveryGroupObject for a list of the data fields that could potentially be returned.



Note: The data fields returned depend on the type of data you ask for in your request.

Syntax

```

deliveryGroupObject[] deliveryGroup = bApi.readDeliveryGroups(filter
    deliveryGroupFilter, pageNumber);

```

readDeliveryGroups Parameters

Name	Type	Required	Description
filter	deliveryGroupFilter	Yes	The filter used to return a specific delivery group.
pageNumber	int	Yes	Retrieves the next “batch” of objects as the value specified increases from 1. In order to obtain an entire set of objects for a given call, you should increase the number value assigned to pageNumber until no more objects are returned.

PHP Code Example

```

<?php
/**
 * This example will return the a deliveryGroupObject for each
 * delivery group that matches the given filter.
 */
$client = new SoapClient('https://api.bronto.com/v4?wsdl', array('trace' =>
    1,
                                'features' => SOAP_SINGLE_ELEMENT_ARRAYS));

try {
    // Add your API token

```

```

$token = "API TOKEN HERE";

print "logging in\n";
$sessionId = $client->login(array('apiToken' => $token))->return;

$session_header = new SoapHeader("http://api.bronto.com/v4",
                                'sessionHeader',
                                array('sessionId' => $sessionId));
$client->__setSoapHeaders(array($session_header));

$filter = array(
    // Filter by name
    "name" => array("operator" => "EqualTo", "value" => "NAME OF DELIVERY
GROUP")
);

$result = $client->readDeliveryGroups(array(
    "filter" => $filter,
    "pageNumber" => 1
));

print_r($result);
} catch (Exception $e) {
    print "uncaught exception\n";
    print_r($e);
}

```

Python Code Example

This example uses the byListType filter.

```

<?php
/**
 * This example will return return the IDs of all
 * deliveries that are part of the delivery group.
 */

$client = new SoapClient('https://api.bronto.com/v4?wsdl', array('trace' =>
1,
                                'features' => SOAP_SINGLE_ELEMENT_ARRAYS));

try {

    // Add your API token
    $token = "API TOKEN HERE";

    print "logging in\n";
    $sessionId = $client->login(array('apiToken' => $token))->return;

    $session_header = new SoapHeader("http://api.bronto.com/v4",
                                    'sessionHeader',
                                    array('sessionId' => $sessionId));
    $client->__setSoapHeaders(array($session_header));

    $filter = array(
        "listByType" => "DELIVERIES",
        "deliveryGroupId" => "ADD THE ID OF THE DELIVERY GROUP"
    );

    $result = $client->readDeliveryGroups(array(

```



```

        "filter" => $filter,
        "pageNumber" => 1
    ));

    print_r($result);
} catch (Exception $e) {
    print "uncaught exception\n";
    print_r($e);
}

```

readDeliveryRecipients

The `readDeliveryRecipients` function returns a `deliveryRecipientStatObject` for each list or segment a delivery was sent to. A `deliveryRecipientStatObject` is also returned for single contact deliveries.

Results

The `readDeliveryRecipients` function may return 1 or many `deliveryRecipientStatObject` objects. For example, if a delivery was sent to 4 lists, a `deliveryRecipientStatObject` will be returned for each list sent to in the delivery (provided you don't filter the request by `listId`). See the documentation on the `deliveryRecipientStatObject` for a list of the data fields that could potentially be returned.



Note: This function does not return a list of each contact who received the delivery. It does return an id for the list or segment the delivery was sent to. If the delivery was only sent to a single contact, a contact id is returned. For more information on result limits and paging, see [How To Read Objects](#).



Note: The data fields returned depend on the type of data you ask for in your request.

Syntax

```

deliveryRecipientStatObject[] deliveries = bApi.readDeliveryRecipients(
    filter deliveryRecipientFilter, pageNumber);

```

readDeliveryRecipients Parameters

Name	Type	Required	Description
<code>filter</code>	deliveryRecipientFilter	Yes	Filter based on a specific delivery and the list, segment, or contact it was sent to.
<code>pageNumber</code>	int	Yes	Retrieves the next “batch” of objects as the value specified increases from 1. In order to obtain an entire set of objects for a given call, you should increase the number value assigned to <code>pageNumber</code> until no more objects are returned.

PHP Code Example

```

<?php
/**
 * This example will return the delivery data for a delivery
 * that was sent to a specific list.

```

```

*/

$client = new SoapClient('https://api.bronto.com/v4?wsdl',
    array('trace' => 1,
        'features' => SOAP_SINGLE_ELEMENT_ARRAYS)
    );

setlocale(LC_ALL, 'en_US');

try {
    // Add in a valid API token
    $token = "VALID API TOKEN";

    print "logging in\n";
    $sessionId = $client->login(array('apiToken' => $token))->return;

    $session_header = new SoapHeader("http://api.bronto.com/v4",
        'sessionHeader',
        array('sessionId' => $sessionId));
    $client->__setSoapHeaders(array($session_header));

    // Set up a filter to return metrics for a delivery sent to a specific
    list.
    // Be sure to replace the generic text below with a valid delivery and
    // list id.
    $filter = array('type' => 'OR',
        'deliveryId' => 'DELIVERYID',
        'listId' => 'LISTID'
    );

    print "Reading delivery data for a delivery sent to a specific list\n";
    $deliveries = $client->readDeliveryRecipients(array('pageNumber' => 1,
        'filter' => $filter,
    ))->return;

    // Print matching results
    foreach ($deliveries as $delivery) {
        print_r($deliveries);
    }
} catch (Exception $e) {
    print "uncaught exception\n";
    print_r($e);
}
?>

```

readFields

The `readFields` function attempts to return fields that match all of the given filter. The specified attributes of the fields are returned for each matching field.

Results

The `readFields` function may return 1 or many field objects. See the documentation on the `fieldObject` for a list of the data fields that could potentially be returned.



Note: The data fields returned depend on the type of data you ask for in your request.

Syntax

```
fieldObject[] fields = bApi.readFields(filter fieldsFilter, pageNumber);
```

readFields Parameters

Name	Type	Required	Comments
filter	fieldsFilter	Yes	The filter used to return only specific fields.
pageNumber	int	Yes	Retrieves the next “batch” of objects as the value specified increases from 1. In order to obtain an entire set of objects for a given call, you should increase the number value assigned to pageNumber until no more objects are returned.
pageSize	int	No	Allows you to set a limit for the page size in order to improve performance speed. If set, the pageSize minimum is 10 and the maximum is 5,000.

PHP Code Example

```

<?php
/**
 * This example will read the details of any fields in your account that
 * have
 * the name 'name' in them (example: firstname, lastname).
 */

$client = new SoapClient('https://api.bronto.com/v4?wsdl', array('trace' =>
    1,
    'features' => SOAP_SINGLE_ELEMENT_ARRAYS));

try {
    $token = "YOUR_TOKEN_HERE";

    print "logging in\n";
    $sessionId = $client->login(array('apiToken' => $token))->return;

    $session_header = new SoapHeader("http://api.bronto.com/v4",
        'sessionHeader',
        array('sessionId' => $sessionId));
    $client->__setSoapHeaders(array($session_header));

    // filter on fields that contain 'name'
    $filter = array('name' => array('operator' => 'Contains',
        'value' => 'name',
        ),
    );

    print "reading all fields that contain the string 'name'\n";
    $fields = $client->readFields(array('pageNumber' => 1,
        'filter' => $filter,
        )
    )->return;

```

```
// print matching results
foreach ($fields as $field) {
    print "Field name: " . $field->name . "; type: " . $field->type . "\n";
}

} catch (Exception $e) {
    print "uncaught exception\n";
    print_r($e);
}
```

readHeaderFooters

The `readHeaderFooters` function attempts to return headers or footers that match the given filter. Header or footer objects can optionally include the associated HTML content.

Results

The `readHeaderFooters` function may return 1 or many header/footer objects. See the documentation on the `headerFooterObject` for a list of the data fields that could potentially be returned.



Note: The data fields returned depend on the type of data you ask for in your request.

Syntax

```
headerFooterObject[] headerFooters = bApi.readHeaderFooters(filter
    headerFooterFilter, includeContent, pageNumber);
```

readHeaderFooters Parameters

Name	Type	Required	Comments
filter	headerFooterFilter	Yes	The filter used to return specific headers or footers.
includeContent	boolean	Yes	Include the HTML portion of returned footers.
pageNumber	int	Yes	Retrieves the next “batch” of objects as the value specified increases from 1. In order to obtain an entire set of objects for a given call, you should increase the number value assigned to <code>pageNumber</code> until no more objects are returned.

PHP Code Example

```
<?php
/**
 * This example will read a list of all available headers and then all
 * footers in your account. It will print the name and the id of each.
 */

$client = new SoapClient('https://api.bronto.com/v4?wsdl', array('trace' =>
    1,
```

```

        'features' => SOAP_SINGLE_ELEMENT_ARRAYS));

try {
    $token = "YOUR_TOKEN_HERE";

    print "logging in\n";
    $sessionId = $client->login(array('apiToken' => $token))->return;

    $session_header = new SoapHeader("http://api.bronto.com/v4",
        'sessionHeader',
        array('sessionId' => $sessionId));
    $client->__setSoapHeaders(array($session_header));

    // get only headers
    $filter = array('position' => 'top');

    print "reading all headers\n";
    $headers = $client->readHeaderFooters(array('pageNumber' => 1,
        'includeContent' => false,
        'filter' => $filter,
        )
    )->return;

    // print matching results
    foreach ($headers as $header) {
        print "Header name: " . $header->name . "; id: " . $header->id . "\n";
    }

    print "\n";

    // get only footers
    $filter = array('position' => 'bottom');

    print "reading all footers\n";
    $headers = $client->readHeaderFooters(array('pageNumber' => 1,
        'includeContent' => false,
        'filter' => $filter,
        )
    )->return;

    // print matching results
    foreach ($headers as $header) {
        print "Footer name: " . $header->name . "; id: " . $header->id . "\n";
    }

} catch (Exception $e) {
    print "uncaught exception\n";
    print_r($e);
}

```

readLists

The `readLists` function attempts to return lists that match all parameters of the given filter. The specific attributes of the lists are returned for each matching list.

Results

The `readLists` function may return 1 or many list objects. See the documentation on the `mailListObject` for a list of the data fields that could potentially be returned.



Note: The data fields returned depend on the type of data you ask for in your request.

Syntax

```
mailListObject[] lists = bApi.readLists(filter mailListFilter, pageNumber);
```

readLists Parameters

Name	Type	Required	Comments
filter	mailListFilter	Yes	The filter used to return specific lists.
pageNumber	int	Yes	Retrieves the next “batch” of objects as the value specified increases from 1. In order to obtain an entire set of objects for a given call, you should increase the number value assigned to pageNumber until no more objects are returned.
pageSize	int	No	Allows you to set a limit for the page size in order to improve performance speed. If set, the pageSize minimum is 10 and the maximum is 5,000.

PHP Code Example

```
<?php
/**
 * This example will return all mailing lists in your account. It will then
 * print each list's name; the number of active contacts on the list; and
 * the list id.
 */

$client = new SoapClient('https://api.bronto.com/v4?wsdl', array('trace' =>
    1,
    'features' => SOAP_SINGLE_ELEMENT_ARRAYS));

try {
    $token = "YOUR_TOKEN_HERE";

    print "logging in\n";
    $sessionId = $client->login(array('apiToken' => $token))->return;

    $session_header = new SoapHeader("http://api.bronto.com/v4",
        'sessionHeader',
        array('sessionId' => $sessionId));
    $client->__setSoapHeaders(array($session_header));

    $filter = array();

    print "reading all lists\n";
    $lists = $client->readLists(array('pageNumber' => 1,
        'filter' => $filter))->return;

    // print matching list names, number of contacts on the list, and ids
```

```

    foreach ($lists as $list) {
        print "Name: " . $list->name . "; contacts: " . $list->activeCount . ";
        id: " . $list->id . "\n";
    }

} catch (Exception $e) {
    print "uncaught exception\n";
    print_r($e);
}

```

readLogins

The `readLogins` function attempts to return the requested data for logins that match the given `loginFilter`.

Results

The `readLogins` function may return 1 or many login objects. See the documentation on the `loginObject` for a list of the data fields that could potentially be returned.



Note: The data fields returned depend on the type of data you ask for in your request.

Syntax

```
loginObject[] logins = bApi.readLogins( filter loginFilter, pageNumber );
```

readLogins Parameters

Name	Type	Required	Description
filter	loginFilter	Yes	The <code>loginFilter</code> allows you to read data from a specific login(s).
pageNumber	int	Yes	Retrieves the next “batch” of objects as the value specified increases from 1. In order to obtain an entire set of objects for a given call, you should increase the number value assigned to <code>pageNumber</code> until no more objects are returned.

PHP Code Example

```

<?php
/**
 * This example will read the details of a login and
 * print out the username and permissions
 * associated with the login.
 */

$client = new SoapClient( 'https://api.bronto.com/v4?wsdl', array( 'trace' =>
    1,
                                'features' => SOAP_SINGLE_ELEMENT_ARRAYS ) );

try {
    $token = "YOUR_TOKEN_HERE";

```

```

print "logging in\n";
$sessionId = $client->login(array('apiToken' => $token))->return;

$session_header = new SoapHeader("http://api.bronto.com/v4",
    'sessionHeader',
    array('sessionId' => $sessionId));
$client->__setSoapHeaders(array($session_header));

$filter = array(
    "type" => "OR",
    "username"=> array("operator" => "StartsWith", "value" => ""),
);

$result = $client->readLogins(array(
    "filter" => $filter,
    "pageNumber" => 1
));

print_r($result);

} catch (Exception $e) {
    print "uncaught exception\n";
    print_r($e);
}

```

readMessageFolders

The `readMessageFolders` function attempts to return message folders that match all of the given filter. The specified attributes of the folders are returned for each matching folder.

Results

The `readMessageFolders` function may return 1 or many message folder objects. See the documentation on the `messageFolderObject` for a list of the data fields that could potentially be returned.



Note: The data fields returned depend on the type of data you ask for in your request.

Syntax

```

messageFolderObject[] folders = bApi.readMessageFolders(filter
    messageFolderFilter, pageNumber);

```

readMessageFolders Parameters

Name	Type	Required	Description
filter	messageFolderFilter	Yes	Used to filter which message object is returned.
pageNumber	int	Yes	Retrieves the next “batch” of objects as the value specified increases from 1. In order to obtain an entire set of objects for a given call, you should increase the number value assigned to <code>pageNumber</code> until no more objects are returned.

PHP Code Example

```

<?php
/**
 * This example will read the message folders from your account and print
 * them out in 'tree-like' fashion.
 */

$client = new SoapClient('https://api.bronto.com/v4?wsdl', array('trace' =>
    1,
                                'features' => SOAP_SINGLE_ELEMENT_ARRAYS));
setlocale(LC_ALL, 'en_US');

try {
    $token = "YOUR_TOKEN_HERE";

    print "logging in\n";
    $sessionId = $client->login(array('apiToken' => $token))->return;

    $session_header = new SoapHeader("http://api.bronto.com/v4",
        'sessionHeader',
        array('sessionId' => $sessionId));
    $client->__setSoapHeaders(array($session_header));

    // get all folders
    $filter = array();

    print "reading message folders\n";
    $folders = $client->readMessageFolders(array('pageNumber' => 1,
        'filter' => $filter,
        )
    )->return;

    // find the root first
    $rootFolder = null;
    foreach($folders as $folder) {
        if (!$folder->parentId) {
            $rootFolder = $folder;
            break;
        }
    }

    // start building up a visual tree view of the folders
    $folderStack = array(array('folder' => $rootFolder,
        'depth' => 0));

    $output = "";
    while (count($folderStack) > 0) {
        $item = array_pop($folderStack);
        $currentFolder = $item['folder'];
        $depth = $item['depth'];
        $indent = str_repeat("    ", $depth);
        $output .= $indent . $currentFolder->name . "\n";
        foreach ($folders as $folder) {
            if ($folder->parentId == $currentFolder->id) {
                array_push($folderStack, array('folder' => $folder, 'depth' =>
                    $depth + 1));
            }
        }
    }

    // print out the tree

```

```

    print $output . "\n";
} catch (Exception $e) {
    print "uncaught exception\n";
    print_r($e);
}

```

readMessageRules

The `readMessageRules` function attempts to return Automated Message Rules that match all of the given filter. The specified attributes of the rules are returned for each matching rule.

Results

The `readMessageRules` function may return 1 or many message rule objects. See the documentation on the `messageRuleObject` for a list of the data fields that could potentially be returned.



Note: The data fields returned depend on the type of data you ask for in your request.

Syntax

```

messageRuleObject[] messageRules = bApi.readMessageRules(filter
    messageRuleFilter, pageNumber);

```

readMessageRules Parameters

Name	Type	Required	Description
filter	messageRuleFilter	Yes	Used to filter which Automated Message Rule objects are returned.
pageNumber	int	Yes	Retrieves the next “batch” of objects as the value specified increases from 1. In order to obtain an entire set of objects for a given call, you should increase the number value assigned to <code>pageNumber</code> until no more objects are returned.

PHP Code Example

```

<?php
/**
 * This example will read the automated message rules from your account
 * that are 'recurring' (i.e. happen on a recurring date). The matching
 * automated message rules will print out their names and ids.
 */

$client = new SoapClient('https://api.bronto.com/v4?wsdl', array('trace' =>
    1,
                                'features' => SOAP_SINGLE_ELEMENT_ARRAYS));
setlocale(LC_ALL, 'en_US');

try {
    $token = "YOUR_TOKEN_HERE";

```

```

print "logging in\n";
$sessionId = $client->login(array('apiToken' => $token))->return;

$session_header = new SoapHeader("http://api.bronto.com/v4",
    'sessionHeader',
    array('sessionId' => $sessionId));
$client->__setSoapHeaders(array($session_header));

// get all rules
$filter = array();

print "reading automated message rules\n";
$rules = $client->readMessageRules(array('pageNumber' => 1,
    'filter' => $filter,
    )
)->return;

// print matching results
foreach ($rules as $rule) {
    if ($rule->type == 'recurring') {
        print "Rule name: " . $rule->name . "; id: " . $rule->id . "\n";
    }
}
} catch (Exception $e) {
    print "uncaught exception\n";
    print_r($e);
}

```

readMessages

The `readMessages` function attempts to return messages that match the given filter. Currently the API does not support the structure of email message editor messages. While you can attempt to read this type of message, the content structure may be incomprehensible.

Results

The `readMessages` function may return 1 or many message objects. See the documentation for `messageObject` and `deliveryObject` for a list of the data fields that could potentially be returned. For more information on result limits and paging, see [How To Read Objects](#).



Note: The data fields returned depend on the type of data you ask for in your request.

Syntax

```

messageObject[] messages = bApi.readMessages(filter messageFilter,
    includeContent, pageNumber);

```

readMessages Parameters

Name	Type	Required	Description
filter	messageFilter	Yes	A filter used for returning specific messages.
includeContent	boolean	Yes	Returns an array of the message content {type, subject, content}

Name	Type	Required	Description
pageNumber	int	Yes	Retrieves the next “batch” of objects as the value specified increases from 1. In order to obtain an entire set of objects for a given call, you should increase the number value assigned to pageNumber until no more objects are returned.
pageSize	int	No	Allows you to set a limit for the page size in order to improve performance speed. If set, the pageSize minimum is 10 and the maximum is 5,000.

PHP Code Example

```

<?php
/**
 * This example will match messages in your account that contain the word
 * 'newsletter'
 * in their name. It will then print out the message names and ids.
 */

$client = new SoapClient('https://api.bronto.com/v4?wsdl', array('trace' =>
    1,
    'features' => SOAP_SINGLE_ELEMENT_ARRAYS));

try {
    $token = "YOUR_TOKEN_HERE";

    print "logging in\n";
    $sessionId = $client->login(array('apiToken' => $token))->return;

    $session_header = new SoapHeader("http://api.bronto.com/v4",
        'sessionHeader',
        array('sessionId' => $sessionId));
    $client->__setSoapHeaders(array($session_header));

    $filter = array('name' => array('operator' => 'Contains',
        'value' => 'newsletter')
    );

    print "reading all matching messages\n";
    $messages = $client->readMessages(array('pageNumber' => 1,
        'includeContent' => false,
        'filter' => $filter))->return;

    // print matching message names and ids
    foreach ($messages as $message) {
        print "Name: " . $message->name . "; id: " . $message->id . "\n";
    }
} catch (Exception $e) {
    print "uncaught exception\n";
    print_r($e);
}

```

```
}
```

readRecentInboundActivities

The `readRecentInboundActivities` function returns a `recentActivityObject` for each activity recorded in your account. You can return up to 30 days worth of data.

Results

The `readRecentInboundActivities` function may return 1 or many `recentActivityObject` objects. See the documentation on the `recentActivityObject` for a list of the data fields that could potentially be returned.



Note: The data fields returned depend on the type of data you ask for in your request.

Syntax

```
recentActivityObject[] activity = bApi.readRecentInboundActivities( filter
    recentInboundActivitySearchRequest);
```

readRecentInboundActivities Parameters

Name	Type	Required	Description
filter	recentInboundActivitySearchRequest	Yes	The <code>recentInboundActivitySearchRequest</code> allows you to specify the specific type of data you want, from a specific time period.

PHP Code Example

```
/**
 * This example will read open and click activity data starting from 30 days
 * ago
 * until the present. It will return 2 pages of results (if that many
 * exist), each
 * page containing 1k recentActivityObjects per page/
 */

$client = new SoapClient('https://api.bronto.com/v4?wsdl', array('trace' =>
    1,
                                                                    'features'
=> SOAP_SINGLE_ELEMENT_ARRAYS));

try {
    $token = "API TOKEN HERE";

    print "logging in\n";
    $sessionId = $client->login(array('apiToken' => $token))->return;

    $session_header = new SoapHeader("http://api.bronto.com/v4",
                                    'sessionHeader',
                                    array('sessionId' => $sessionId));
    $client->__setSoapHeaders(array($session_header));

    // compute date/time 30 days ago
    $startDate = date('c', strtotime('-30 days'));
```

```

// First Page
$filter = array(
    "start" => $startDate,
    "size" => "1000",
    "types" => array("click", "open"),
    "readDirection" => 'FIRST',
);

print "reading activity details page 1: \n";
$activities = $client->readRecentInboundActivities(array(
    'filter' => $filter,
));

$i = 0;
foreach ($activities as $activity) {
    if ($i == 0) {
        print("recentActivityObject #: " . $i . "\n");
        print_r($activity);
    }
    $i++;
}

print "Request:\n" . $client->__last_request . "\n";

// Second Page
$filter = array(
    "start" => $startDate,
    "size" => "1000",
    "types" => array("open", "click"),
    "readDirection" => 'NEXT',
);

print "reading activity details page 2 \n";
$activities = $client->readRecentInboundActivities(array(
    'filter' => $filter,
));

foreach ($activities as $activity) {
    if ($i == 1000) {
        print("recentActivityObject #: " . $i . "\n");
        print_r($activity);
    }
    $i++;
}

print "Request:\n" . $client->__last_request . "\n";
} catch (Exception $e) {
    print "uncaught exception\n";
    print_r($e);
}

```

Python Code Example

```

from suds.client import Client
from suds import WebFault
import sys
import logging
from datetime import datetime, timedelta

```

```

# Tested with Python 2.6.1 and suds soap library version 0.4

# See suds home page:
# https://fedorahosted.org/suds/

# Bronto API WSDL
BRONTO_WSDL = 'https://api.bronto.com/v4?wsdl'

# start up basic logging
logging.basicConfig()

TOKEN = "API TOKEN HERE"

# login using the token to obtain a session ID
bApi = Client( BRONTO_WSDL )

try:
    # Use an existing session ID if you have one, otherwise, login
    # and obtain a new session ID
    # session_id = ""
    session_id = bApi.service.login(TOKEN)
# exit if something goes wrong
except WebFault, e:
    print '\nERROR MESSAGE: '
    print e
    sys.exit()

# Set up the soap headers using the
# session_id obtained from login()
session_header = bApi.factory.create("sessionHeader")
session_header.sessionId = session_id
bApi.set_options(soapheaders=session_header)

# Create the recentInboundActivitySearchRequest passed into
# readRecentInboundActivities()

filter = bApi.factory.create('recentInboundActivitySearchRequest')
readDirection = bApi.factory.create('readDirection')
readDirection = "FIRST"

# Read data starting from 30 days ago up to now
filter.start = datetime.now() + timedelta(-30)
filter.size = 1000
filter.readDirection = readDirection
# Only return data for opens and clicks
filter.types = ['click', 'open']

# Initialize our counters
i = 1
j = 0
# Only get 5 pages worth of data
while i <= 5:
    if i == 1:
        print "Reading data for page 1 \n"
        try:
            read_activity = bApi.service.readRecentInboundActivities(filter)
        except WebFault, e:
            print '\nERROR MESSAGE: '
            print e
            sys.exit()

```

```

else:
    print "Reading data for page " + str(i) + "\n"
    filter.readDirection = 'NEXT'
    try:
        read_activity = bApi.service.readRecentInboundActivities(filter)
    except WebFault, e:
        print '\nERROR MESSAGE: '
        print e
        print "No data on page " + str(i)
        sys.exit()
    i = i + 1
    for activity in read_activity:
        print "recentActivityObject: " + str(j)
        print activity
        j = j + 1

```

readRecentOutboundActivities

The `readRecentOutboundActivities` function returns a `recentActivityObject` for each activity recorded in your account. You can return up to 1 day worth of data.

Results

The `readRecentOutboundActivities` function may return 1 or many `recentActivityObject` objects. See the documentation on the `recentActivityObject` for a list of the data fields that could potentially be returned.



Note: The data fields returned depend on the type of data you ask for in your request.

Syntax

```

recentActivityObject[] activity = bApi.readRecentOutboundActivities( filter
    recentOutboundActivitySearchRequest);

```

readAccounts Parameters

Name	Type	Required	Description
filter	<code>recentOutboundActivitySearchRequest</code>	Yes	The <code>recentOutboundActivitySearchRequest</code> allows you to specify the specific type of data you want, from a specific time period.

PHP Code Example

```

/**
 * This example will read send activity data starting from 10 hours ago
 * until the present. It will return 2 pages of results (if that many
 * exist), each
 * page containing 1k recentActivityObjects per page/
 */

$client = new SoapClient('https://api.bronto.com/v4?wsdl', array('trace' =>
    1,
                                                                    'features'
=> SOAP_SINGLE_ELEMENT_ARRAYS));

```



```

try {
    $token = "API TOKEN HERE";

    print "logging in\n";
    $sessionId = $client->login(array('apiToken' => $token))->return;

    $session_header = new SoapHeader("http://api.bronto.com/v4",
                                     'sessionHeader',
                                     array('sessionId' => $sessionId));
    $client->__setSoapHeaders(array($session_header));

    // compute date/time 10 hours ago
    $startDate = date('c', strtotime('-10 hours'));

    // First Page
    $filter = array(
        "start" => $startDate,
        "size" => "1000",
        "types" => array("send"),
        "readDirection" => 'FIRST',
    );

    print "reading activity details page 1: \n";
    $activities = $client->readRecentOutboundActivities(array(
        'filter' => $filter,
    ));

    $i = 0;
    foreach ($activities as $activity) {
        if ($i == 0) {
            print("recentActivityObject #: " . $i . "\n");
            print_r($activity);
        }
        $i++;
    }
    print "Request:\n" . $client->__last_request . "\n";

    // Second Page
    $filter = array(
        "start" => $startDate,
        "size" => "1000",
        "types" => array("send"),
        "readDirection" => 'NEXT',
    );

    print "reading activity details page 2 \n";
    $activities = $client->readRecentOutboundActivities(array(
        'filter' => $filter,
    ));

    foreach ($activities as $activity) {
        if ($i == 1000) {
            print("recentActivityObject #: " . $i . "\n");
            print_r($activity);
        }
        $i++;
    }
    print "Request:\n" . $client->__last_request . "\n";
} catch (Exception $e) {

```

```

        print "uncaught exception\n";
        print_r($e);
    }

```

Python Code Example

```

from suds.client import Client
from suds import WebFault
import sys
import logging
from datetime import datetime, timedelta

# Tested with Python 2.6.1 and suds soap library version 0.4

# See suds home page:
# https://fedorahosted.org/suds/

# Bronto API WSDL
BRONTO_WSDL = 'https://api.bronto.com/v4?wsdl'

# start up basic logging
logging.basicConfig()

TOKEN = "API TOKEN HEE"

# login using the token to obtain a session ID
bApi = Client( BRONTO_WSDL )

try:
    # Use an existing session ID if you have one, otherwise, login
    # and obtain a new session ID
    # session_id = ""
    # session_id = bApi.service.login(TOKEN)
    # exit if something goes wrong
except WebFault, e:
    print '\nERROR MESSAGE: '
    print e
    sys.exit()

# Set up the soap headers using the
# session_id obtained from login()
session_header = bApi.factory.create("sessionHeader")
session_header.sessionId = session_id
bApi.set_options(soapheaders=session_header)

# Create the recentOutboundActivitySearchRequest passed into
# readRecentOutboundActivities()

filter = bApi.factory.create('recentOutboundActivitySearchRequest')
readDirection = bApi.factory.create('readDirection')
readDirection = "FIRST"

# Read data starting from 10 hours ago up to now
filter.start = datetime.now() + timedelta(-10)
filter.size = 1000
filter.readDirection = readDirection
# Only return data for sends
filter.types = ['send']

# Initialize our counters

```

```

i = 1
j = 0
# Only get 5 pages worth of data
while i <= 5:
    if i == 1:
        print "Reading data for page 1 \n"
        try:
            read_activity =
bApi.service.readRecentOutboundActivities(filter)
        except WebFault, e:
            print '\nERROR MESSAGE: '
            print e
            sys.exit()
    else:
        print "Reading data for page " + str(i) + "\n"
        filter.readDirection = 'NEXT'
        try:
            read_activity =
bApi.service.readRecentOutboundActivities(filter)
        except WebFault, e:
            print '\nERROR MESSAGE: '
            print e
            print "No data on page " + str(i)
            sys.exit()
    i = i + 1
    for activity in read_activity:
        print "recentActivityObject: " + str(j)
        print activity
        j = j + 1

```

readSegments

The readSegments function attempts to return segments that match all of the given filter. The specified attributes of the segments are returned for each matching segment.

Results

The readSegments function may return 1 or many segment objects. See the documentation on the segmentObject for a list of the data fields that could potentially be returned.



Note: The data fields returned depend on the type of data you ask for in your request.

Syntax

```

segmentObject[] segments = bApi.readSegments( filter segmentFilter,
pageNumber);

```

readSegment Parameters

Name	Type	Required	Description
filter	segmentFilter	Yes	Used to filter which segment objects are returned.

Name	Type	Required	Description
pageNumber	int	Yes	Retrieves the next “batch” of objects as the value specified increases from 1. In order to obtain an entire set of objects for a given call, you should increase the number value assigned to pageNumber until no more objects are returned.
pageSize	int	No	Allows you to set a limit for the page size in order to improve performance speed. If set, the pageSize minimum is 10 and the maximum is 5,000.

PHP Code Example

```

<?php
/**
 * This example will read all segments from your account that contain
 * the word 'test' as a part of their name. It will then print out
 * the name of the segment and its id.
 */

$client = new SoapClient('https://api.bronto.com/v4?wsdl', array('trace' =>
    1,
                                'features' => SOAP_SINGLE_ELEMENT_ARRAYS));
setlocale(LC_ALL, 'en_US');

try {
    $token = "YOUR_TOKEN_HERE";

    print "logging in\n";
    $sessionId = $client->login(array('apiToken' => $token))->return;

    $session_header = new SoapHeader("http://api.bronto.com/v4",
        'sessionHeader',
        array('sessionId' => $sessionId));
    $client->__setSoapHeaders(array($session_header));

    // get all segments that contain the word 'test'
    $filter = array('name' => array('operator' => 'Contains',
        'value' => 'test'
    ));

    print "reading segments\n";
    $segments = $client->readSegments(array('pageNumber' => 1,
        'filter' => $filter,
    ))->return;

    // print matching results
    foreach ($segments as $segment) {
        print "Segment name: " . $segment->name . "; id: " . $segment->
        id . "\n";
    }
}

```

```

    }

    } catch (Exception $e) {
        print "uncaught exception\n";
        print_r($e);
    }

```

readSMSDeliveries

The `readSMSDeliveries` function attempts to return SMS deliveries that match all of the given filter. An SMS delivery consists of an SMS message that has been sent to a specific set of recipients. Although an SMS delivery is associated with an SMS message, the SMS message's content may have changed since the SMS delivery was sent and should not be used for determining the actual content sent. Instead, the content that was delivered may be returned by this function.

Results

The `readSMSDeliveries` function may return 1 or many `smsDeliveryObjects`. For more information on result limits and paging, see [How To Read Objects](#).



Note: The data fields returned depend on the type of data you ask for in your request.

Syntax

```

smsDeliveryObject[] deliveries = bApi.readSMSDeliveries( filter
    smsDeliveryFilter includeRecipients, includeContent, pageNumber);

```

readDeliveries Parameters

Name	Type	Required	Description
filter	smsDeliveryFilter	Yes	A filter used for returning specific SMS deliveries.
includeRecipients	boolean	Yes	Include the recipients of returned SMS deliveries. You will get back an array of the recipients who were, or are scheduled to receive the SMS delivery. If the SMS delivery was sent to a keyword, you will get back a keyword id.
includeContent	boolean	Yes	Include the content of returned SMS deliveries.
pageNumber	int	Yes	Retrieves the next “batch” of objects as the value specified increases from 1. In order to obtain an entire set of objects for a given call, you should increase the number value assigned to pageNumber until no more objects are returned.

readSMSKeywords

The `readSMSKeywords` function attempts to return keywords that match all of the given filters.

Results

The `readSMSKeywords` function may return 1 or many `smsKeyword` objects. See the documentation on the `smsKeywordObject` for a list of the data fields that could potentially be returned.



Note: The data fields returned depend on the type of data you ask for in your request.

Syntax

```
smsKeywordObject[] keywords = bApi.readSMSKeywords( filter smsKeywordFilter,
    includeDeleted, pageNumber);
```

readSMSKeywords Parameters

Name	Type	Required	Description
<code>filter</code>	smsKeywordFilter	Yes	The filter used to return a specific SMS keyword.
<code>includeDeleted</code>	boolean	No	If set to true, SMS keywords marked as deleted will be returned. When you delete an SMS keyword, it is marked for deletion and then deleted 7 days later.
<code>pageNumber</code>	int	Yes	Retrieves the next “batch” of objects as the value specified increases from 1. In order to obtain an entire set of objects for a given call, you should increase the number value assigned to <code>pageNumber</code> until no more objects are returned.

readSMSMessages

The `readSMSMessages` function attempts to return SMS messages that match the given filter.

Results

The `readSMSMessages` function may return 1 or many SMS message objects. See the documentation on the `smsMessageObject` for a list of the data fields that could potentially be returned. For more information on result limits and paging, see [How To Read Objects](#).



Note: The data fields returned depend on the type of data you ask for in your request.

Syntax

```
smsMessageObject[] messages = bApi.readSMSMessages(filter messageFilter,
    includeContent, pageNumber);
```

readSMSMessage Attributes

Name	Type	Required	Description
filter	messageFilter	Yes	A filter used for returning specific SMS messages.
includeContent	boolean	Yes	Returns a string of the SMS message content .
pageNumber	int	Yes	Retrieves the next “batch” of objects as the value specified increases from 1. In order to obtain an entire set of objects for a given call, you should increase the number value assigned to pageNumber until no more objects are returned.

readUnsubscribes

The `readUnsubscribes` function attempts to return unsubscribe data for a contact or a delivery.

Results

The `readUnsubscribes` function may return 1 or many unsubscribe objects. See the documentation on the `unsubscribeObject` to view the data that could potentially be returned.

Syntax

```
unsubscribeObject[] unsubscribes = bApi.readUnsubscribes (filter
    unsubscribeFilter, pageNumber);
```

readLists Parameters

Name	Type	Required	Comments
filter	unsubscribeFilter	Yes	The filter used to return specific unsubscribe data.
pageNumber	int	Yes	Retrieves the next “batch” of objects as the value specified increases from 1. In order to obtain an entire set of objects for a given call, you should increase the number value assigned to pageNumber until no more objects are returned.

PHP Code Example

```
<?php
/**
 * This will return all unsubscribes associated with an account.
 * You can filter what unsubscribes are returned using the unsubscribeFilter
 * @see ../reference/r_api_soap_unsubscribefilter/
```

```

*
* @param int $pageNumber
*     Returns the next bath of objects as you increase the value from 1
* @param unsubscribeFilter $filter
*     The filter used to return specific unsubscribe data
*
* @return array An array containing unsubscribe objects
* @see ../reference/r_api_soap_general_unsubscribeobject/
*
* @copyright Copyright (c) 2018 Oracle + Bronto
*/

$client = new SoapClient('https://api.bronto.com/v4?wsdl', array('trace' =>
1,
                                'features' => SOAP_SINGLE_ELEMENT_ARRAYS));

try {
    $token = "YOUR_API_KEY";

    print "logging in\n";
    $sessionId = $client->login(array('apiToken' => $token))->return;

    $session_header = new SoapHeader("http://api.bronto.com/v4",
        'sessionHeader',
        array('sessionId' => $sessionId));
    $client->__setSoapHeaders(array($session_header));

    // No filter set
    $filter = array();

    // Example filter by ContactId
    //$filter = array("contactId" => "CONTACT_ID");

    print "reading all lists\n";
    $unsubscribes = $client->readUnsubscribes(array('pageNumber' => 1, 'filter'
=> $filter))->return;

    print_r($unsubscribes);

    foreach ($unsubscribes as $unsubscribe) {
        // Print contactId, method, and date unsubscribed
        print "contactId: " . $unsubscribe->contactId . "; method: " .
$unsubscribe->method . "; created: " . $unsubscribe->created . "\n";
    }
} catch (Exception $e) {
    print "uncaught exception\n";
    print_r($e);
}
?>

```

readWebforms

The readWebforms function attempts to return webforms that match all of the given filters.

Results

The readWebforms function may return 1 or many webform objects. See the documentation on the webformObject for a list of the data fields that could potentially be returned.

Syntax

```
webformObject[] webforms = bApi.readWebforms( filter webformFilter,
    pageNumber);
```

readWebforms Parameters

Name	Type	Required	Description
filter	webformFilter	Yes	The filter used to return specific webforms.
pageNumber	int	Yes	Retrieves the next “batch” of objects as the value specified increases from 1. In order to obtain an entire set of objects for a given call, you should increase the number value assigned to pageNumber until no more objects are returned.

PHP Code Example

```
<?php
/**
 * This example will return webform objects for webforms
 * that whose name matched the given filter.
 */

$client = new SoapClient('https://api.bronto.com/v4?wsdl',
    array('trace' => 1,
        'features' => SOAP_SINGLE_ELEMENT_ARRAYS)
);

try {
    // Add in a valid API token
    $token = "ADD TOKEN HERE";

    print "logging in\n";
    $sessionId = $client->login(array('apiToken' => $token))->return;

    $session_header = new SoapHeader("http://api.bronto.com/v4",
        'sessionHeader',
        array('sessionId' => $sessionId));
    $client->__setSoapHeaders(array($session_header));
    // set up a filter to read webforms
    $filter = array('type' => 'AND',
        'name' => array('operator' => 'Contains',
            'value' => 'SOME VALUE')
    );

    print "reading webforms\n";
    $webforms = $client->readWebforms(array('pageNumber' => 1,
        'filter' => $filter
    ))->return;
```

```

} catch (Exception $e) {
    print "uncaught exception\n";
    print $e;
    exit();
}

// print matching webforms
foreach ($webforms as $webform) {
    print("ID: " . $webform->id . " Name: " . $webform->name . "\n");
}
?>

```

Python Code Example

```

from suds.client import Client
from suds import WebFault
import sys
import logging

# Tested with Python 2.6.1 and suds soap library version 0.4

# See suds home page:
# https://fedorahosted.org/suds/

# Bronto API WSDL
BRONTO_WSDL = 'https://api.bronto.com/v4?wsdl'

# start up basic logging
logging.basicConfig()

TOKEN = "TOKEN HERE"

# login using the token to obtain a session ID
bApi = Client( BRONTO_WSDL )

try:
    session_id = bApi.service.login(TOKEN)
    # just exit if something goes wrong
except WebFault, e:
    print '\nERROR MESSAGE: '
    print e
    sys.exit()

# Set up the soap headers using the
# session_id obtained from login()
session_header = bApi.factory.create("sessionHeader")
session_header.sessionId = session_id
bApi.set_options(soapheaders=session_header)

# Create the webformFilter passed into
# readWebforms()

filter = bApi.factory.create('webformFilter')

stringValue = bApi.factory.create('stringValue')
stringValue.value = 'SOME VALUE'
filterOperator = bApi.factory.create('filterOperator')
stringValue.operator = filterOperator.Contains

```

```

filterType = bApi.factory.create('filterType')

filter.name = stringValue
filter.type = filterType.AND

pageNumber = 1

try:
    webform_list = bApi.service.readWebforms(filter, pageNumber)
except WebFault, e:
    print '\nERROR MESSAGE: '
    print e
    sys.exit()
print bApi.last_sent()
print bApi.last_received()
print webform_list
for webform in webform_list:
    print 'Webform Name: ' + webform.name
    print 'Webform ID: ' + webform.id
    print 'Webform Type: ' + webform.type

```

readWorkflows

The `readWorkflows` function attempts to return the requested data for workflows that match the given `workflowFilter`.

Results

The `readWorkflows` function may return 1 or many workflow objects. See the documentation on the `workflowObject` for a list of the data fields that could potentially be returned.



Note: The data fields returned depend on the type of data you ask for in your request.

Syntax

```

workflowObject[] workflow = bApi.readWorkflows( filter workflowFilter,
    pageNumber );

```

readAccounts Parameters

Name	Type	Required	Description
filter	workflowFilter	Yes	The <code>workflowFilter</code> allows you to read data from a specific workflow(s).
pageNumber	int	Yes	Retrieves the next “batch” of objects as the value specified increases from 1. In order to obtain an entire set of objects for a given call, you should increase the number value assigned to <code>pageNumber</code> until no more objects are returned.

PHP Code Example

```
<?php
/**
 * This example will read data for a workflow
 * whose name matches the given filter
 */

$client = new SoapClient('https://api.bronto.com/v4?wsdl',
    array('trace' => 1,
        'features' => SOAP_SINGLE_ELEMENT_ARRAYS)
    );

try {

    // Add in a valid API token here
    $token = "ADD IN A VALID API TOKEN";

    print "logging in\n";
    $sessionId = $client->login(array('apiToken' => $token))->return;

    $session_header = new SoapHeader("http://api.bronto.com/v4",
        'sessionHeader',
        array('sessionId' => $sessionId)
    );

    $client->__setSoapHeaders(array($session_header));

    // Replace the generic text with name of the workflow
    // whose data you want to obtain.
    $filter = array('name' => array('operator' => 'EqualTo',
        'value' => 'NAME OF A WORKFLOW'))
    );

    print "reading activity details\n";

    $workflows = $client->readWorkflows(array('pageNumber' => 1,
        'filter' => $filter,)
    );

    // Print the data for the workflow object(s) returned
    foreach ($workflows as $workflow) {
        print_r($workflows);
    }

} catch (Exception $e) {
    print "uncaught exception\n";
    print_r($e);
}
```

Results of Add, Update, and Delete Functions

Add, update, and delete functions return a writeResult object. This object contains two arrays: results and errors. The results array will contain N resultItem objects, where N is the number of objects you provided for the function.

Each resultItem object has the following elements:

- id: the id of the element, if it already existed at Bronto
- isNew: a boolean indicating whether a new object was created or not.
- isError: a boolean indicating whether or not a problem occurred when calling the function with this object.
- errorCode: a numeric code for the type of error, if one occurred while calling the function with this object.

- `errorString`: a description of an error, if one occurred while calling the function with this object.

The order of the results array will match the order of the objects you originally provided to the function, so you can map back the `resultItem` objects to your own objects to take any further action you require.

As an additional convenience, if there were any errors, a second array called `errors` will also be provided. This array will contain integers indicating the index in the results array of all `resultItem` objects that had errors, so if you do not wish to iterate over the entire results array, you can quickly pick out any problems. A complete description of all error codes and messages is available.

SOAP Filter Objects

Filter are objects that you can apply to your SOAP calls. They are used to target a specific subset of the object you are working with.

Bronto's Filter topics include both definitions of object filters and the components that define them.

accountFilter Multi-Brand Only

The `accountFilter` is used to target a specific subset of `accountObjects` during an your SOAP call. The `readAccounts` page includes an example of `accountFilter`.

Name	Type	Required	Description
<code>type</code>	filterType	Required if more than one filter is being defined. Defaults to AND if no <code>type</code> is specified.	Used to define multiple filters.
<code>id</code>	<code>string[]</code>	No	The unique id of the account you want to match.
<code>name[]</code>	stringValue[]	No	Allows you to match accounts based on criteria involving the name of the account.

activityFilter

The `activityFilter` is used to filter based on activity.

Name	Type	Required	Description
<code>start</code>	<code>dateValue</code>	Yes	The date you want to start reading activity date from.
<code>size</code>	<code>int</code>	Yes	The number of activity objects to be returned.
<code>types</code>	<code>stringValue[]</code>	No	The type of activity data you want to return. Valid values are open , click , adconv , bounce , send , unsubscribe , or view . You can return more than one type at a time.

Name	Type	Required	Description
readDirection	stringValue[]	No, defaults to FIRST	<p>Allows you to page through results returned. Page sizes are limited to results of 1000 to 5000, depending on what you specify for the <code>size</code>. Valid values are FIRST and NEXT.</p> <p>NOTE: Starting a new query with NEXT will throw an error. You must start with FIRST.</p>

apiTokenFilter

Name	Type	Required	Description
id	string[]	No	The unique id of the API token you want to match.
accountId	string[]	No	The unique id of the account you want to match with an API token.
name[]	stringValue[]	No	Allows you to match API tokens based on criteria involving the name.

bounceFilter

Name	Type	Required	Description
contactId	string	Yes	The ID of the contact you want to read bounce data for.
start	dateTime	Yes	The date you want to start reading activity data from. The start date can be up to 30 days in the past.
end	dateTime	No	The date you want to stop reading from. If you want to read from the start date up to now, do not include this property in your request.

contactFilter

Name	Type	Required	Description
type	filterType	Required if more than one filter is being defined. Defaults to AND if no type is specified.	Used to define multiple filters.

Name	Type	Required	Description
email	stringValue []	No	Allows filtering of contacts based on matching email addresses.
mobileNumber	stringValue []	No	Allows filtering of contacts based on matching mobile numbers. The only supported filter operator for mobileNumber is <code>EqualTo</code> . If you use a different filter operator, you will still see results returned for the <code>EqualTo</code> operator.
id	string []	No	Allows filtering of contacts based on id. You can obtain the id for a contact by calling readContacts , or by looking at the footer when viewing the overview page for an individual contact in the application.
status	string []	No	Allows filtering of contacts based on a status, including <code>active</code> , <code>onboarding</code> , <code>bounce</code> , <code>unsub</code> , <code>unconfirmed</code> , and <code>transactional</code> .
created	dateValue []	No	Allows filtering of contacts based on criteria involving the creation date and time of the contact.
modified	dateValue []	No	Allows filtering of contacts based on criteria involving the last modification date and time of the contact.
listId	string []	No	Allows filtering of contacts based on an id of a list they belong to.
segmentId	string []	No	Allows filtering of contacts based on an id of a segment they belong to.
SMSKeywordID	string []	No	Allows filtering of contacts based on an id of an SMS keyword they are subscribed to.
msgPref	string []	No	Allows filtering of contacts based on their message format preference (<code>html</code> or <code>text</code>).

Name	Type	Required	Description
source	string[]	No	Allows filtering on the source of the contact (manual, import, api, webform, or sforcereport.)
customSource	stringValue[]	No	Allows you to filter on the custom source provided when the contact was imported or signed up.

contentTagFilter

Name	Type	Required	Description
type	filterType	Required if more than one filter is being defined. Defaults to AND if no type is specified.	Used to define multiple filters.
id	string[]	No	The unique id of the content tag you want to read data from.
name	stringValue[]	No	Allows you to read data from a content tag based on the content tag's name.

conversionFilter

You can specify one filter key per request. For example, you can perform a request for two `contactIds`, in which case, conversions for both `contactIds` should be returned. You can not specify both a `contactId` and `deliveryId`.

Name	Type	Required	Description
contactId	string[]	No	An array of contacts, referenced by id, to read conversion data for.
deliveryId	string[]	No	An array of deliveries, referenced by id, to read conversion data for.
id	string[]	No	An array of ids assigned to conversions you want to read data for.
orderId	string[]	No	An array of orders, referenced by id, you want read conversion data for.

dateValue

Name	Type	Required	Description
value	dateTime	Yes	The value which the operator will act on in order to filter which object(s) will be returned. Format should be YYYY-MM-DDThh:mm:ssTZD (e.g. 2009-07-16T19:20:30-05:00)
operator	filterOperator	Yes	The operator used to determine how the object is filtered.


deliveryFilter


Name	Type	Required	Description
type	filterType	Not required but defaults to AND if no type is specified.	Defines whether all start dateValue filters must match the delivery or if a single match is sufficient.
id	string[]	No	The unique id of the specific delivery you want to read.
messageId	string[]	No	The unique id of the message you want to read deliveries for.
start	dateValue[]	No	<p>Allows you to read data for deliveries based on their send date. In the deliveryFilter, the value you set for the start li represents the send date of the delivery.</p> <ul style="list-style-type: none"> • See the example code below to see how to read deliveries made in a specific date range. • If you want to read data for deliveries that were made outside of the data retention policy, include archived in the status array.

Name	Type	Required	Description
status	string[]	No	The status of this delivery: <ul style="list-style-type: none"> • sent • sending • unsent • archived • skipped • tmp
deliveryType	string[]	No	The type of delivery: <ul style="list-style-type: none"> • normal - A regular email delivery sent via the application GUI. • test - A test delivery. • automated - A delivery sent via an Automator or Automated Message Rule. • split - A delivery made as part of an A/B split test. • transactional - Transactional delivery • triggered - API triggered delivery.

deliveryGroupFilter

The deliveryGroupFilter can be used to select a subset of delivery groups. The readDeliveryGroups overview includes an example of using a filterOperator to limit the number of delivery groups returned.

Name	Type	Required	Description
type	filterType	Required if more than one filter is being defined. Defaults to AND if no type is specified.	Used to define multiple filters.
name	stringValue[]	No	The name associated with the delivery group. <div>  Note: For readDeliveryGroups, only the EqualTo and StartsWith filterOperators are supported. </div>

Name	Type	Required	Description
listByType	string	No	Valid values are DELIVERIES , AUTOMATORS , or MESSAGEGROUPS . You need to pass in “listByType” in the filter section if you want to tell the API which arrays to return.  Note: listByType should only be used if you want data on specific types of items (messages, deliveries, automated message rules) added to a particular delivery group. When using listByType, you should always identify the specific delivery group whose items you want to read data from using deliveryGroupId.
deliveryGroupId	string	No	The unique id assigned to a delivery group. This will return data for a delivery group that matches the specific id.
deliveryId	string	No	The unique id assigned to a specific delivery. This will return data for all delivery groups containing this delivery.
messageGroupId	string	No	The unique id assigned to a specific message group. This will return data for all delivery groups containing this message.
automatorId	string	No	The unique id assigned to an automated message rule. This will return data for all delivery groups containing this automated message rule.

deliveryRecipientFilter

If you only pass in a deliveryId, a deliveryRecipientStatObject will be returned for each list or segment the delivery was sent. If the delivery was only sent to a single contact, a deliveryRecipientStatObject will be returned for the single contact delivery.

Name	Type	Required	Description
type	filterType	Required if more than one filter is being defined. Defaults to AND if no type is specified.	Used to define multiple filters.
deliveryId	string	Yes	The unique id of the delivery you want to read.
listIds[]	string or an array of strings	No	The unique id of a list or lists sent to in the delivery. This allows you to filter the request so only deliveryRecipientStatObjects for the specified lists are returned.

Name	Type	Required	Description
segmentIds []	string or an array of strings	No	The unique id of a segment or segments sent to in the delivery. This allows you to filter the request so only deliveryRecipientStatObjects for the specified segments are returned.
contactIds []	string or an array of strings	No	The unique id for a contact or contacts. Test deliveries can be sent to multiple single contacts. In this case, a deliveryRecipientStatObject is returned for each contact, because each send is considered a single contact delivery. You can filter the request to only return deliveryRecipientStatObjects for the specified contactIds.

fieldsFilter

Name	Type	Required	Description
type	filterType	Required if more than one filter is being defined. Defaults to AND if no type is specified.	Used to define multiple filters.
id	string[]	No	The unique id of the field you want to read data from.
name	stringValue []	No	Allows you to read data from a field based on criteria involving the internal name of the field.

filterOperators

Valid Filter Operators	Value	Description
EqualTo / NotEqualTo	string	Used to match if a value used in the object filter equals or does not equal the value set for the filter.
StartsWith / EndsWith	string	Used to determine if the value used in the object filter starts with or ends with the string value set for the filter.
DoesNotStartWith / DoesNotEndWith	string	Used to determine if the name of the object does start with or does not end with the string value set for the filter.
GreaterThan / LessThan	string	The operator used to determine how the object is filtered.

Valid Filter Operators	Value	Description
GreaterThanOrEqualTo / LessThanOrEqualTo	string	The operator used to determine how the object is filtered.
Contains / DoesNotContain	string	The operator used to determine how the object is filtered.
SameYear / NotSameYear	string	The operator used to determine how the object is filtered. Used specifically for <code>dateValue</code> values.
SameDay / NotSameDay	string	The operator used to determine how the object is filtered. Used specifically for <code>dateValue</code> values.
Before / After	string	The operator used to determine how the object is filtered. Used specifically for <code>dateValue</code> values.
BeforeOrSameDay / AfterOrSameDay	string	The operator used to determine how the object is filtered. Used specifically for <code>dateValue</code> values.

filterType

Name	Type	Required	Description
AND	string	No. You must, however, use OR if AND is not used.	Used to define multiple filters for an object or objects that match each filter.
OR	string	No. You must, however, use AND if OR is not used.	Used to define multiple filters for an object or objects that match one filter or another.

headerFooterFilter

Name	Type	Required	Description
type	<code>filterType[]</code>	Required if more than one filter is being defined. Defaults to AND if no type is specified.	Used to define multiple filters.
id	string[]	No	The unique id of a header or footer object to match.
name	<code>stringValue[]</code>	No	Allows you to match specific headers or footers by name.
position	string[]	No	Can be either <code>top</code> to return headers, or <code>bottom</code> to return footers.

loginFilter

Name	Type	Required	Description
type	filterType	Required if more than one filter is being defined. Defaults to AND if no type is specified.	Used to define multiple filters.
username	stringValue[]	No	Allows filtering of logins based on their username.

mailListFilter

Name	Type	Required	Description
type	filterType	Required if more than one filter is being defined. Defaults to AND if no type is specified.	Used to define multiple filters.
id	string[]	No	The unique id of the list you want to read data from. You can obtain the id for a list by calling readLists , or by looking at the footer when viewing the overview page for an individual list in the application.
name	stringValue[]	No	Allows you to read data from a list based on criteria involving the name of the list.

messageFilter

Name	Type	Required	Description
type	filterType	Required if more than one filter is being defined. Defaults to AND if no type is specified.	
id	string[]	No	The unique id of the message you want to match. You can obtain the id for a message by calling readMessages , or by looking at the footer when viewing the overview page for an individual message in the application.
name	stringValue[]	No	Allows you to match messages based on criteria involving the name of the message.
status	string[]	No	

Name	Type	Required	Description
messageFolderId	string[]	No	Allows you to match messages based on the id of the message folder where the message resides.


messageFolderFilter

Name	Type	Required	Description
type	filterType	Required if more than one filter is being defined. Defaults to AND if no type is specified.	Used to define multiple filters.
id	string[]	No	The unique id of the account you want to read data from.
name	stringValue[]	No	Allows you to match folders based on criteria involving the name of the folder.
parentid	string[]	No	The unique id of the parent folder you want to filter for matching folders.

messageRuleFilter

Name	Type	Required	Description
type	filterType	Required if more than one filter is being defined. Defaults to AND if no type is specified.	Used to define multiple filters.
id	string[]	No	The unique id of the Automated Message Rule you want to match.
name	stringValue[]	No	Allows you to match Automated Message Rules based on criteria involving the name of the Automated Message Rule.
ruleType	string[]	No	Allows you to match Automated Message Rules of only a specific type. Type can be one of activity, date, recurring, or api.

readDirection

Name	Type	Required	Description
readDirection	string	No. Defaults to <code>FIRST</code> if no value is provided	<p>Allows you to page through results returned. Page sizes are limited to results of 1000 or 5000, depending on what you specify for the <code>size</code>. Valid values are:</p> <ul style="list-style-type: none"> <code>FIRST</code> <code>NEXT</code> <p> Note: Starting a new query with <code>NEXT</code> will throw an error. You must start with <code>FIRST</code>.</p>

recentInboundActivitySearchRequest

The values in this table can be used as filters when making a `readRecentInboundActivities` or `readRecentOutboundActivities` call. `Start`, `size`, and `readDirection` are required for these calls.

Name	Type	Required	Description
types	string	No	<p>The type of inbound activity you want to get. Valid values are:</p> <ul style="list-style-type: none"> <code>bounce</code> <code>contactSkip</code> <code>open</code> <code>click</code> <code>conversion</code> <code>reply</code> <code>unsubscribe</code> <code>friendforward</code> <code>social</code> <code>webform</code> <code>sms_bounce</code> <code>sms_reply</code>
start	dateTime	Yes	The date you want to start reading activity data from. The start date can be up to 30 days in the past.
end	dateTime	No	The date you want to stop reading from. If you want to read from the start date up to now, do not include this li in your request.
contactId	string	No	The ID of a contact you want to read activity data for.

Name	Type	Required	Description
deliveryId	string	No	The ID of a delivery you want to read activity data for.
size	int	Yes	The number of objects returned. Valid values are any number between 1000 and 5000
readDirection	readDirection	Yes	see readDirection

recentOutboundActivitySearchRequest

Name	Type	Required	Description
types	string	No	The type of outbound activity you want to get. Valid values are: <ul style="list-style-type: none"> • send • sms_send
start	dateTime	Yes	The date you want to start reading activity data from. The start date can be up to 30 days in the past.
end	dateTime	No	The date you want to stop reading from. The end date should be set to within 24 hours of the start date. If you set an end date that is greater than 24 hours, only the activity from the first 24 hours is returned. If you want to view the activity for multiple days, you will need to make multiple API calls.
contactId	string	No	The ID of a contact you want to read activity data for.
deliveryId	string	No	The ID of a delivery you want to read activity data for.
size	int	Yes	The number of objects returned. Valid values are any number between 1000 and 5000
readDirection	readDirection	Yes	see readDirection

segmentFilter

Name	Type	Required	Description
type	filterType	Required if more than one filter is being defined. Defaults to AND if no type is specified.	Used to define multiple filters.
id	string[]	No	The unique id of the segment you want to match. You can obtain the id for a segment by calling readSegments , or by looking at the footer when viewing the overview page for an individual segment in the application.
name	stringValue[]	No	Allows you to match segments based on criteria involving the name of the segment.

smsDeliveryFilter

Name	Type	Required	Description
type	filterType	Required if more than one filter is being defined. Defaults to AND if no type is specified.	Used to define multiple filters.
id	string[]	No	The unique ids assigned to SMS deliveries you want to read.
messageId	string[]	No	The unique id of SMS message you want to read SMS deliveries for.
start	dateValue[]	No	Allows you to read data for SMS deliveries based on their send date. In the <code>smsDeliveryFilter</code> , the value you set for the <code>start</code> li represents the send date of the SMS delivery.
status	string[]	No	The status of the SMS delivery: <ul style="list-style-type: none"> • sent • sending • unsent • archived • skipped • tmp

Name	Type	Required	Description
deliveryType	string[]	No	<p>The deliveryType indicates the how the SMS delivery was made. Valid values are:</p> <ul style="list-style-type: none"> • bulk • test • workflow • transaction

smsKeywordFilter

Name	Type	Required	Description
type	filterType	Required if more than one filter is being defined. Defaults to AND if no type is specified.	Used to define multiple filters.
id	string[]	No	The unique id assigned to an SMS keyword.
name	stringValue []	No	Allows you to match messages based on criteria involving the name of the message.

Name	Type	Required	Description
keywordType	string[]	Yes	<p>The type set for the SMS keyword. Valid values are:</p> <ul style="list-style-type: none"> • <code>basic</code> – Basic keywords are non-subscription keywords meant for individual transactions. With basic keywords, a person texts into a keyword and a response is sent back. The interaction ends there. The recipient is not added to a list because they have not agreed to receive future marketing messages from you. • <code>subscription</code> – Subscription based keywords require a person to choose to receive SMS messages from you by texting into a given keyword. Contacts who subscribe to a subscription based keyword will be added to a list so that you can send SMS messages to them in the future. • <code>text2join</code> – <code>text2join</code> keywords allows you to grow your list by providing a way for potential contacts to text their email address in and automatically be added to one of your lists.

stringValue

Name	Type	Required	Description
value	string	Yes	The value which the with which the operator will act on in order to filter which object(s) will be returned.
operator	filterOperator	Yes	The operator used to determine how the object is filtered.

unsubscribeFilter

Name	Type	Required	Description
contactId	string	No	The ID of a contact you want to read unsubscribe data for.
deliveryId	string	No	The ID of a delivery you want to read unsubscribe data for.
start	dateTime	No	The date you want to start reading activity data from. The start date can be up to 30 days in the past.
end	dateTime	No	The date you want to stop reading from. If you want to read from the start date up to now, do not include this property in your request.

webformFilter

Name	Type	Required	Description
type	filterType	Required if more than one filter is being defined. Defaults to AND if no type is specified.	Used to define multiple filters.
id	string[]	No	Allows filtering of webforms based on id
name	stringValue []	No	Allows filtering of webforms based on the name assigned to webforms.
webformType	string[]	No	Allows filtering of webforms based on type. Valid values are: <ul style="list-style-type: none"> • managePreferences • addContact • forwardToAFriend • lookupContact • thankYou • confirmation • unsubscribe • complaint

workflowFilter

Name	Type	Required	Description
type	filterType	Required if more than one filter is being defined. Defaults to AND if no type is specified.	Used to define multiple filters.
id	string[]	No	The unique id of the workflow you want to read data for.
name	stringValue[]	No	Allows you to read workflows based on criteria involving the name of the workflow.

SOAP Settings Objects

Settings objects are specific to modifying the Bronto platform's account settings.

accountAllocations


The `accountAllocations` settings let you allocate features to accounts.


Overview


You can allocate things like emails, inbox previews, and additional users. Depending on the function you are calling (`addAccounts` or `updateAccounts`), different allocation settings will be required. Make sure you view the appropriate Required column for the function you want to call below. Calling `readAccounts` will return all of the allocations settings associated with the account object.


Changing an account's allocating features in excess of the defaults provided by the bundle type (only Advanced accounts can purchase add-ons) will cause extra costs to be incurred. Please be mindful of these additional charges when calling `addAccounts` or `updateAccounts`. For details on the cost you may accrue, contact your Account Manager.

Name	Required for addAccounts Only	Required for updateAccounts Agency	Type	Description
canExceedAllocation	Yes	No	boolean	Allows the account to exceed their allocated emails if set to 1.
canExceedSmsAllocation	No	No	boolean	Allows the account to exceed their allocated SMS if set to 1.
emails	Yes	No	long	The number of emails allocated to the account.

Name	Required for addAccounts Agency Only	Required for updateAccounts	Type	Description
contacts	Yes	No	long	<p>For advanced account types only. Used if you want to add-on to the default contact allocation and allocate more than the default 1,000,000 contacts. You can specify as many contacts as you like, but keep in mind that contacts are billed in bundles of 1M (1,000,000). For example, if you specify 2,000,001 contacts in the call, you will be billed for 3M contacts and receive 3M additional contacts.</p> <p> Note: – This will cause additional costs to be incurred. Contact your account manager for pricing information.</p>

Name	Required for addAccountsAgency Only	Required for updateAccounts	Type	Description
hosting	Yes	No	long	<p>For advanced account types only. Used if you want to add-on to the default hosting allocation and allocate more than the default 100 MB of hosting space. You must refer to the amount of hosting you would like to add in bytes (100MB = 104857600 bytes). You can specify as much hosting as you like, but keep in mind that hosting is billed in bundles of 200MB. For example, if you specify 210,763,776 (201MB) of hosting in the call, you will be billed for 400MB of hosting and receive 400MB of additional hosting, thus bringing your total to 501MB of hosting.</p> <p> Note: – This will cause additional costs to be incurred. Contact your account manager for pricing information.</p>

Name	Required for addAccounts Agency Only	Required for updateAccounts	Type	Description
logins	Yes	No	long	<p>For advanced account types only. Used if you want to add-on to the default users allocation and allocate more than the default 5 users for the account. You can purchase as many users as you like, but keep in mind that users are billed in bundles of 10. For example, if you specify 11 users in the call, you will be billed for 20 users and receive 20 additional users.</p> <p> Note: – This will cause additional costs to be incurred. Contact your account manager for pricing information.</p>
api	Yes	No	boolean	Gives API access to the account if set to 1.

Name	Required for addAccounts Agency Only	Required for updateAccounts	Type	Description
fields	Yes	No	long	<p>For advanced account types only. Used if you want to add-on to the default fields allocation and allocate more than the default 100 fields. You can purchase as many fields as you like, but keep in mind that fields are billed in bundles of 100. For example, if you specify 201 fields in the call, you will be billed for 300 fields and receive 300 additional fields.</p> <p> Note: – This will cause additional costs to be incurred. Contact your account manager for pricing information.</p>
startDate	Yes	No	dateTime	<p>The date that the allocations begin on. Format should be YYYY-MM-DDThh:mm:ssTZD (e.g. 2009-07-16T19:20:30-05:00)</p>
periodFrequency	Yes	Yes	int	<p>The allocation period length in months (i.e. How much time until the allocations are replenished). The valid values are 1, 3, 6, and 12.</p>

Name	Required for addAccounts Agency Only	Required for updateAccounts	Type	Description
bundle	Yes	Yes	string	<p>The type of account you are adding. The valid values are <code>professional</code> and <code>core</code>. For legacy agency accounts, you will need to use <code>basic</code>, <code>intermediate</code>, and <code>advanced</code>.</p> <ul style="list-style-type: none"> • <code>professional</code> – Provides access to core email marketing features as well as an advanced feature set which includes: Automated Message Rules, Conversion Tracking, Direct Update, Connections, A/B splits, Advanced Reporting, and Dynamic Content. Professional accounts receive 5 users, 100MB of hosting, 100 fields, 1,000,000 contacts, 12 Inbox Previews, and API Access. Add-ons may also be purchased for Professional accounts, allowing you to further increase any of these allocations. • <code>core</code> – Provides access to core email marketing features. Core accounts receive 5 users, 100MB of hosting, 100 fields, and 1,000,000 contacts. <p>Legacy</p> <ul style="list-style-type: none"> • <code>Basic</code> – Provides access to core

Name	Required for addAccounts Agency Only	Required for updateAccounts	Type	Description
defaultTemplates	Yes	No	boolean	Whether not the sub-account has access to the default templates we provide.

contactInformation

The contact information settings are important because you must provide your company or organization name and full postal mailing address in every message in order to be CAN-SPAM compliant.

Name	Type	Description
organization	string	The organization name set for the account.
firstName	string	The first name, used for contact information, set for the account.
lastName	string	The last name, used for contact information, set for the account.
email	string	The contact email address set for the account.
phone	string	The telephone number set for the account. The country code is required.
address	string	The primary address set for the account.
address2	string	The secondary address set for the account.
city	string	The city set for the account.
state	string	The state set for the account. Use two character abbreviations for states. For example, for North Carolina, use NC.
zip	string	The zip code set for the account.
country	string	The country set for the account. You must use a two letter country code. A list of country codes can be found here .
notes (Agency Only)	string	Any notes added by the agency administrator for the client account.

Country Codes

When adding or updating an account, you must use a two letter country code when specifying the country. A list of countries and their corresponding two letter code are listed below.

Country Code	Country
AF	Afghanistan
AX	Aland Islands

Country Code	Country
AL	Albania
DZ	Algeria
AS	American Samoa
AD	Andorra
AO	Angola
AI	Anguilla
AQ	Antarctica
AG	Antigua and Barbuda
AR	Argentina
AM	Armenia
AW	Aruba
AU	Australia
AT	Austria
AZ	Azerbaijan
BS	Bahamas
BH	Bahrain
BD	Bangladesh
BB	Barbados
BY	Belarus
BE	Belgium
BZ	Belize
BJ	Benin
BM	Bermuda
BT	Bhutan
BO	Bolivia Plurinational State of
BA	Bosnia and Herzegovina
BW	Botswana
BV	Bouvet Island
BR	Brazil
IO	British Indian Ocean Territory
BN	Brunei Darussalam
BG	Bulgaria
BF	Burkina Faso
BI	Burundi

Country Code	Country
KH	Cambodia
CM	Cameroon
CA	Canada
CV	Cape Verde
KY	Cayman Islands
CF	Central African Republic
TD	Chad
CL	Chile
CN	China
CX	Christmas Island
CC	Cocos (keeling) Islands
CO	Colombia
KM	Comoros
CG	Congo
CD	Congo the Democratic Republic of the
CK	Cook Islands
CR	Costa Rica
CI	“Cote dIvoire”
HR	Croatia
CU	Cuba
CY	Cyprus
CZ	Czech Republic
DK	Denmark
DJ	Djibouti
DM	Dominica
DO	Dominican Republic
EC	Ecuador
EG	Egypt
SV	El Salvador
GQ	Equatorial Guinea
ER	Eritrea
EE	Estonia
ET	Ethiopia
FK	Falkland Islands (malvinas)

Country Code	Country
FO	Faroe Islands
FJ	Fiji
FI	Finland
FR	France
GF	French Guiana
PF	French Polynesia
TF	French Southern Territories
GA	Gabon
GM	Gambia
GE	Georgia
DE	Germany
GH	Ghana
GI	Gibraltar
GR	Greece
GL	Greenland
GD	Grenada
GP	Guadeloupe
GU	Guam
GT	Guatemala
GG	Guernsey
GN	Guinea
GW	Guinea-bissau
GY	Guyana
HT	Haiti
HM	Heard Island and Mcdonald Islands
VA	Holy See (Vatican City State)
HN	Honduras
HK	Hong Kong
HU	Hungary
IS	Iceland
IN	India
ID	Indonesia
IR	Iran Islamic Republic of
IQ	Iraq

Country Code	Country
IE	Ireland
IM	Isle of Man
IL	Israel
IT	Italy
JM	Jamaica
JP	Japan
JE	Jersey
JO	Jordan
KZ	Kazakhstan
KE	Kenya
KI	Kiribati
KP	“Korea Democratic Peoples Republic of”
KR	Korea Republic of
KW	Kuwait
KG	Kyrgyzstan
LA	“Lao Peoples Democratic Republic”
LV	Latvia
LB	Lebanon
LS	Lesotho
LR	Liberia
LY	Libyan Arab Jamahiriya
LI	Liechtenstein
LT	Lithuania
LU	Luxembourg
MO	Macao
MK	Macedonia the Former Yugoslav Republic of
MG	Madagascar
MW	Malawi
MY	Malaysia
MV	Maldives
ML	Mali
MT	Malta
MH	Marshall Islands
MQ	Martinique

Country Code	Country
MR	Mauritania
MU	Mauritius
YT	Mayotte
MX	Mexico
FM	Micronesia Federated States of
MD	Moldova Republic of
MC	Monaco
MN	Mongolia
ME	Montenegro
MS	Montserrat
MA	Morocco
MZ	Mozambique
MM	Myanmar
NA	Namibia
NR	Nauru
NP	Nepal
NL	Netherlands
AN	Netherlands Antilles
NC	New Caledonia
NZ	New Zealand
NI	Nicaragua
NE	Niger
NG	Nigeria
NU	Niue
NF	Norfolk Island
MP	Northern Mariana Islands
NO	Norway
OM	Oman
OT	Other
PK	Pakistan
PW	Palau
PS	Palestinian Territory Occupied
PA	Panama
PG	Papua New Guinea

Country Code	Country
PY	Paraguay
PE	Peru
PH	Philippines
PN	Pitcairn
PL	Poland
PT	Portugal
PR	Puerto Rico
QA	Qatar
RE	REunion
RO	Romania
RU	Russian Federation
RW	Rwanda
BL	Saint Barthélemy
SH	Saint Helena Ascension and Tristan da Cunha
KN	Saint Kitts and Nevis
LC	Saint Lucia
MF	Saint Martin
PM	Saint Pierre and Miquelon
VC	Saint Vincent and the Grenadines
WS	Samoa
SM	San Marino
ST	Sao Tome and Principe
SA	Saudi Arabia
SN	Senegal
RS	Serbia
SC	Seychelles
SL	Sierra Leone
SG	Singapore
SK	Slovakia
SI	Slovenia
SB	Solomon Islands
SO	Somalia
ZA	South Africa
GS	South Georgia and the South Sandwich Islands

Country Code	Country
ES	Spain
LK	Sri Lanka
SD	Sudan
SR	Suriname
SJ	Svalbard and Jan Mayen
SZ	Swaziland
SE	Sweden
CH	Switzerland
SY	Syrian Arab Republic
TW	Taiwan Province of China
TJ	Tajikistan
TZ	Tanzania United Republic of
TH	Thailand
TL	Timor-leste
TG	Togo
TK	Tokelau
TO	Tonga
TT	Trinidad and Tobago
TN	Tunisia
TR	Turkey
TM	Turkmenistan
TC	Turks and Caicos Islands
TV	Tuvalu
UG	Uganda
UA	Ukraine
AE	United Arab Emirates
GB	United Kingdom
US	United States
UM	United States Minor Outlying Islands
UY	Uruguay
UZ	Uzbekistan
VU	Vanuatu
VE	Venezuela Bolivarian Republic of
VN	Viet Nam

Country Code	Country
VG	Virgin Islands British
VI	Virgin Islands U.S.
WF	Wallis and Futuna
EH	Western Sahara
YE	Yemen
ZM	Zambia
ZW	Zimbabwe

formatSettings

All the times in the account are dependent on the time zone settings you specify for the form. For example, if you set the time zone setting to `America/New_York`, then all time dependent settings in the account (such as message send times) will be set according to `America/New_York` time.

Name	Type	Description
timeZone	string	The time zone used for the account. For example: <code>EST</code> .
dateFormat	date	The date format used for the account For example: <code>yyyy-mm-dd</code> .
locale	string	The locale used for the account. For example: <code>en_US</code> .

generalSettings

The general settings let you define the general settings for an account. Depending on the function you are calling (`addAccounts` or `updateAccounts`), different allocation settings will be required. Make sure you view the appropriate Required column for the function you want to call below. Calling `readAccounts` will return all of the allocations settings associated with the account object.

Name	Required for addAccounts Only	Required for updateAccounts	Type	Description
emergencyEmail	Yes	No	string	The emergency contact email address for the account. This address will be used for notification purposes if a message is unable to be sent due to it exceeding the email allocation. Please use an address which is constantly monitored by some one at your organization.

Name	Required for addAccounts Agency Only	Required for updateAccounts	Type	Description
bounceLimit	Yes	Yes	long	The bounce limit set for the account. The bounce limit represents the number of times contacts can consecutively bounce without recorded activity before they are automatically unsubscribed.
dailyFrequencyCap	No	No	long	The maximum number of emails an individual contact can receive in a single day from this account.
weeklyFrequencyCap	No	No	long	The maximum number of emails an individual contact can receive in a single week from this account.
monthlyFrequencyCap	No	No	long	The maximum number of emails an individual contact can receive in a 30 day period from this account.
textDelivery	No	No	boolean	If set to 1, when a test delivery is made, two different deliveries will actually get created. One delivery will have the regular (HTML) version of the message. The other delivery will be the plain text version. Each delivery will be treated separately for accounting/tracking purposes.

Name	Required for addAccounts Agency Only	Required for updateAccounts	Type	Description
textPreference	No	No	boolean	If set to 1, adds a checkbox to your Webform or Manage Preferences page that lets your contacts choose between receiving text only or HTML messages.
useSSL	No	No	boolean	If set to 1, this will encrypt the accounts session using SSL.
currentContacts	No	No	long	The total numbers of contacts currently created in the account.
maxContacts	No	No	long	The maximum number of contacts allowed in the account for the current allocation period.
currentMonthlyEmails	No	No	long	The total number of emails sent this month from the account.
currentHostingSize	No	No	long	The current number of hosting space being used by the account.
maxHostingSize	No	No	long	The total amount of hosting space allotted to the account.
sendUsageAlerts	Yes	Agency Only Yes	boolean	Sends an email when the account exceeds the allocation thresholds if set to 1.
usageAlertEmail	Yes	Agency Only Yes	string	The email address that will receive usage alert notifications.
agencyTemplateuploadPerm	No	Agency Only Yes	boolean	Only for updateAccounts and readAccounts . Allows the account to upload custom templates if set to 1.

Name	Required for addAccounts Agency Only	Required for updateAccounts	Type	Description
defaultTemplates	Yes	Agency Only Yes	boolean	Only for updateAccounts and readAccounts . Allows the account to access the default templates we provide if set to 1.
enableInboxPreviews	Yes	Agency Only Yes	boolean	Only for updateAccounts and readAccounts . Gives the account the ability to purchase inbox previews if set to 1.
allowCustomizedBranding	Yes	Agency Only Yes	boolean	Only for updateAccounts and readAccounts . Allows the account to change the powered-by logo used in messages if set to 1.

readOnlyContactData

The readOnlyContactData object contains read-only contact level data returned when calling readContacts.

Name	Type	Comments
geoIPCity	string	The city recorded for the contact based on their last known non-mobile IP addresses.
geoIPStateRegion	string	The state/region recorded for the contact based on their last known non-mobile IP addresses.
geoIPZip	string	The zip code recorded for the contact based on their last known non-mobile IP addresses.
geoIPCountry	string	The country recorded for the contact based on their last known non-mobile IP addresses.
geoIPCountryCode	string	The country code recorded for the contact based on their last known non-mobile IP addresses.
primaryBrowser	string	The primary browser (Firefox, Chrome, Safari, etc.) used by a contact.
mobileBrowser	string	The mobile browser (Safari mobile, Firefox mobile, Chrome mobile) used by a contact.

Name	Type	Comments
primaryEmailClient	string	The primary email client (Microsoft Outlook, Mozilla Thunderbird, Apple Mail, etc.) used by a contact.
mobileEmailClient	string	The mobile email client (Gmail mobile, Yahoo Mail for mobile, etc.) used by a contact.
operatingSystem	string	The operating system (MacOSX, WinXP, Win7, Android, iOS etc.) used by a contact.
firstOrderDate	dateTime	The date of the first order recorded for a contact.
lastOrderDate	dateTime	The date of the last order recorded for a contact.
lastOrderTotal	decimal	The total amount of revenue recorded for the most recent order.
totalOrders	long	The total number of orders recorded for a contact.
totalRevenue	decimal	The total amount of revenue recorded for a contact.
averageOrderValue	decimal	The average amount of revenue per order recorded for a contact.
lastDeliveryDate	dateTime	The last date a delivery was made to the contact.
lastOpenDate	dateTime	The last date an open was recorded for the contact.
lastClickDate	dateTime	The last date a click was recorded for the contact.

repliesSettings

The following settings are applied to email replies that are handled by the application when reply tracking is enabled for a delivery. Reply classification, similar to spam filtering, cannot perfectly identify all emails. Before enabling any of these options you should send several deliveries to ensure that the automatic classifications are accurate for your messages.

Name	Type	Description
deletedAutomatedReplies	boolean	Handles automated replies, such as vacation replies.
deleteSpam	boolean	Deletes replies determined by the application to be spam.
handleUnsubscribes	boolean	Unsubscribes contacts whose replies match key words set in the unsubscribeKeywords option.

Name	Type	Description
unsubscribeKeywords	string	Keywords (comma separated) used to classify replies as unsubscribe requests. Used in conjunction with handleUnsubscribes.
replyForwardEmail	string	Replies that are not caught by the above filters will be forwarded to these addresses (comma separated).
deleteUnsubscribeReplies	boolean	Deletes replies determined to be unsubscribe requests.

SOAP General Objects

Here we define the objects you can interact with using Bronto's SOAP API. This includes general, filter, and error objects.

accountObject Multi-Brand Only

The account object contains information about the account associated with the API token you authenticated with. Information includes the name of the account, various settings and information such as contact information, sending allocation, frequency cap settings, time zone and locale settings, branding settings, etc.

Name	Type	Description
id	string	The unique id assigned to the account.
name	string	The name assigned to the account.
status	string	The status of the account. You can update an account from unrestricted to inactive and inactive to unrestricted. You can't change the status of a restricted account. Users will not be able to login (from the UI or the API) to accounts with a status of inactive.
generalSettings	GeneralSettings[]	An array of the general settings set for the account. Click in the Type column for more information.
contactInformation	ContactInformation[]	An array of the contact information set for the account. Click in the Type column for more information.
formatSettings	FormatSettings[]	An array of the formatting settings set for the account. Click in the Type columns for more information.
repliesSettings	RepliesSettings[]	An array of the replies settings set for the account. Click in the Type columns for more information.

Name	Type	Description
allocations	AccountAllocations[]	An array of values corresponding to the allocations you assign to the account you are adding. Click in the Name row for more information on the items in the array.

activityObject

The activity object contains activity data about contacts, messages, and deliveries.

Name	Type	Description
activityDate	dateTime	The date the activity occurred on.
contactId	string	The ID assigned to the contact associated with the activity.
deliveryId	string	The ID assigned to the delivery associated with the activity.
messageId	string	The ID assigned to the message associated with the activity.
listId	string	The ID assigned to the list that the delivery associated with the activity was sent to.
segmentId	string	The ID assigned to the segment that the delivery associated with the activity was sent to.
trackingType	string	The type of activity the object represents. The trackingType can be: <ul style="list-style-type: none"> • open • click • adconv • bounce • send • unsubscribe • view
bounceReason	string	For bounce activities returned, the detailed reason why the bounce occurred.

Name	Type	Description
bounceType	string	For bounce activities returned, the type of bounce recorded. The following types can be returned for bounceType: Hard Bounces <ul style="list-style-type: none"> • conn_perm • sub_perm • content_perm Soft Bounces <ul style="list-style-type: none"> • conn_temp • sub_temp • content_temp • other
linkName	string	For click activities returned, the name of the link that was clicked.
linkUrl	string	For click activities returned, the URL of the link that was clicked.

apiTokenObject

An API token is used to manage API access. You can grant basic permissions (read, write, and send), as well as activate and deactivate specific tokens.

Name	Type	Comments
id	string	The unique id for the API token. The id can be used to reference a specific API token when using the apiToken functions.
name	string	The name assigned to API token. The name can be used to reference a specific API token when using the apiToken functions.
permissions	int	The permissions assigned to the API token. An API token can have read, write, and send permissions. Each permission is assigned an int value. To assign: <ul style="list-style-type: none"> • Read = 1 • Write = 2 • Read, Write = 3 • Send = 4 • Read, Send = 5 • Send, Write = 6 • Read, Write, Send = 7

Name	Type	Comments
active	boolean	Whether or not the API token is active. You can always go back and activate or deactivate API tokens at a later time.
created	string	Tells you when the API token was created.
modified	string	Tells you when the API token was last modified.
accountId	string	The account, referenced by ID, that the API token is assigned to.

bounceObject

The bounce object contains data about an email bounce recorded in your account.

Name	Type	Description
contactId	string	The ID fo the contact associated with the bounce.
deliveryId	string	The ID of the delivery associated with the bounce.
type	string	The type of bounce. The following types can be returned: Hard Bounces <ul style="list-style-type: none"> • conn_perm • sub_perm • content_perm Soft Bounces <ul style="list-style-type: none"> • conn_temp • sub_temp • content_temp • other
description	string	A description of the bounce.
created	dateTime	The date and time the bounce was recorded.

contactField

Any fielddata you use in a message must be smaller than 1 MB.

Name	Type	Description
fieldId	string	The id of the field you want to update. You can obtain a field id by calling readFields .

Name	Type	Description
content	anyType	<p>The data you want to add to the field. They type of data you add to the field should match the type set for the field (text, date, integer, etc.).</p> <p>The type of field:</p> <ul style="list-style-type: none"> • <code>text</code> – Text box (max character limit: 65535) • <code>textarea</code> – A large multi-line text box (max character limit: 65535) • <code>password</code> – A text box that hides typed characters (max character limit: 65535) • <code>checkbox</code> – A boolean value of true (checked) or false (unchecked). Accepts a string value of <code>true</code> or <code>false</code> • <code>radio</code> – Radio buttons. Accepts values that are predefined in the UI or a value that is passed in, if it is not already defined, but it will not be saved as a new option. • <code>select</code> – Pull-down menu • <code>integer</code> – Accepts any 10-digit, whole number less than or equal to 2147483647. If you need to store a static number (i.e. a number that is not incrementing or decrementing) we suggest using either a predefined field, or a custom field with a type of Text. For example, to store telephone numbers use one of the predefined fields created for storing telephone numbers. If you are storing incrementing or decrementing numbers in a custom field created with a type of Number, be mindful of the 10 digit, 2147483647 limit. • <code>currency</code> – Accepts any positive or negative number with two decimal points (max character limit: 15) • <code>float</code> – Accepts any positive or negative number with a decimal point (max character limit: 53) • <code>date</code> – A value that matches a specific date. See ../reference/r_api_soap_datevalue.xml for more information on the date format.

contactObject

A contact describes an individual email address and/or SMS number in Bronto, along with associated statistics and field data that you have provided.

Name	Type	Comments
id	string	The unique id for the contact. The id can be used to reference a specific contact when using the contact functions. You can obtain the id for a contact by calling readContacts , or by looking at the footer when viewing the overview page for an individual contact in the application.
email	string	The email address assigned to the contact. The email address can be used to reference a specific contact when using the contact functions.
mobileNumber	string	The mobile number stored for the contact. A valid country code must be included when adding or updating a mobile number for a contact. The phone number should not include any punctuation or spaces (Valid example: 19195551212).

Name	Type	Comments
status	string	<p>The status of the contact. Each valid status is described below:</p> <ul style="list-style-type: none"> • <code>active</code> – Active contacts are “live” contacts that can receive both marketing and transactional emails from you. • <code>onboarding</code> – onboarding contacts are new contacts who are waiting to be sent their first marketing email from you. Once they are sent the first marketing email, they will enter the assessment phase of the automated onboarding process and automatically move from onboarding to Awaiting Assessment, and then to <code>active</code>. You can not update a contact’s status if they currently have a status of onboarding. Contacts with a status of onboarding can receive transactional emails from you. • <code>transactional</code> – Contacts with a status of <code>transactional</code> can only be sent transactional emails. A transactional email facilitates an agreed-upon transaction, or updates a customer in an existing business relationship. • <code>bounce</code> – The bounce contact status indicates that you cannot send messages to this contact because they have received a hard bounce due to bad email address, or they have exceed the bounce limit for your account. • <code>unconfirmed</code> – Contacts with a status of <code>unconfirmed</code> have not yet agreed to receive your marketing email messages. Remember, one of our policies is that contacts must agree to receive your marketing email messages in order for you to send marketing emails to them. Contacts with a status of <code>unconfirmed</code> can receive transactional emails from you. • <code>unsub</code> – Contacts with a status of <code>unsub</code> have either unsubscribed themselves from receiving marketing email messages from you, or were unsubscribed by you. Contacts with a status of <code>unsub</code> can

Name	Type	Comments
msgPref	string	The message preference for the contact. A contact can have a message preference of text or html.
source	string	The source or where the contact came from. The source can manual, import, api, webform, or sfcoreport (salesforce report).
customSource	string	A source you define that states where the contact came from.
created	dateTime	The date the contact was created. This timestamp is immutable and cannot be changed
modified	dateTime	The last time information about the contact was modified. This timestamp is immutable and cannot be changed.
deleted	boolean	Set to true if the contact has been deleted.
listIds	string, array. Use an array for multiple ids	The lists (referenced by ID) that the contact belongs to. You can obtain listIds by calling the readLists function.
fields	contactField[]	Fields and corresponding field data associated with the contact.
SMSKeywordIDs	string, array. Use an array for multiple ids	An array of the ids corresponding to SMS keywords the contact is subscribed to.
numSends	long	The total number of deliveries sent to the contact.
numBounces	long	The total number of times deliveries sent to the contact resulted in a bounce.
numOpens	long	The total number of times deliveries were opened by the contact. This metric includes multiple opens of the same delivery.
numClicks	long	The total number of times deliveries were clicked by the contact. If a link is clicked multiple times, each click is included in this metric.
numConversions	long	The total number of conversions made by the contact.
conversionAmount	float	The sum/total amount of conversions made by the contact.

Name	Type	Comments
<code>readOnlyContactData</code>	<code>readOnlyContactData[]</code>	An object containing read-only contact level data returned when calling <code>readContacts</code> .

contentTagObject

Content tags allow you to create reusable blocks of content that you can use in the body, header, and footer of your email messages. The block is referenced via a custom defined content tag you create. When the message is sent, the content tag is replaced with the appropriate content.



Name	Type	Comments
<code>id</code>	string	The unique id for the content tag.
<code>name</code>	string	<p>The name you assigned to the content tag. The name you specify will be used to reference this block of content via the Content Tag. For example, if you name the Content Tag <code>mycontenttag</code>, you would reference this Content Tag in your message by adding <code>%@mycontenttag%</code> to your message. Note:</p> <ul style="list-style-type: none"> • The content tag name must be 100 characters or less. • The name cannot be blank or null
<code>value</code>	string	<p>The content that will be displayed when the message is sent. Note:</p> <ul style="list-style-type: none"> • The value cannot contain other content tags, field tags, API message tags, or dynamic code. • HTML can be used in the value, however, the HTML will appear unformatted if the content tag is used in a plain text message.

conversionObject

A conversion is used track when a contact performs an action in response to an email you send them (make a purchase, download a whitepaper, etc.).

Name	Type	Comments
<code>id</code>	string	The unique id assigned to the conversion
<code>contactId</code>	string	The unique id assigned to the contact you want to associate a conversion with.
<code>email</code>	string	The email address of the conversion contact.

Name	Type	Comments
orderId	string	The order identifier. This should be unique per order and will be used to prevent duplicate orders.
item	string	The SKU of the line item. Certain item codes are reserved for special use: subtotal , taxes , shipping , and total .
description	string	The description of the line item.
quantity	int	The unit count of the line item.
amount	decimal	The line item total amount of the line item.
orderTotal	int	The total number of orders made
createdDate	dateTime	The date and time of the conversion. If no date/time is provided, then the system will timestamp the record.
deliveryId	string	The unique id assigned to a delivery you want to associate a conversion with.
messageId	string	The unique id assigned to a message you want to associate a conversion with.
automatorId	string	The unique id assigned to an automator you want to associate a conversion with.
listId	string	The unique id assigned to a list you want to associate a conversion with.
segmentId	string	The unique id assigned to a segment you want to associate a conversion with.

Name	Type	Comments
deliveryType	string	<p>If the conversion object is associated with a delivery, the type of delivery will be returned when calling readConversion. Valid values are:</p> <ul style="list-style-type: none"> • <code>normal</code> – A regular email delivery sent via the application GUI. • <code>test</code> – A test delivery. • <code>automated</code> – A delivery sent via an Automator or Automated Message Rule. • <code>split</code> – A delivery made as part of an A/B split test. • <code>transactional</code> – Transactional delivery • <code>triggered</code> – API triggered delivery. <p>If you are on the old order service and you try to pass a type that is not one of the 4 valid types then type will be ignored.</p>
tid	string	<p>A unique id which associates a conversion with a specific contact and delivery. When conversion tracking is enabled in the application, you can pull the <code>tid</code> from the bronto tracking cookie and use it in the <code>addConversion</code> call to record a conversion that a specific contact makes from a specific delivery.</p> <p> Note: The <code>tid</code> will take priority over other ids (<code>deliveryId</code>, <code>contactId</code>, <code>email</code>) passed in.</p> <p> Note: The <code>tid</code> is not returned when calling readConversion. In order to obtain the <code>tid</code>, you must pull it from the bronto tracking cookie.</p>

deliveryGroupObject

The delivery group object contains information about delivery groups in your account.

Name	Type	Description
id	string	The unique id assigned to the delivery group.

Name	Type	Description
name	string	The name assigned to the delivery group.
deliveryCount	long	The total number of deliveries currently contained in the delivery group.
createdDate	dateTime	The date the delivery group was created.
deliveryIds[]	string	An array of the ids for each delivery contained in the delivery group.
messageRuleIds[]	string	An array of the ids for each automated message rule contained in the delivery group.
messageIds[]	string	An array of the ids for each message contained in the delivery group.
numSends	long	The number of times this delivery has been sent.
numDeliveries	long	The number of successful deliveries.
numBounces	long	The total number of bounces recorded for the delivery.
uniqOpens	long	The number of unique opens recorded for the delivery.
numOpens	long	The total number of opens recorded for the delivery.
avgOpens	long	The number of average opens recorded for the delivery.
uniqClicks	long	The number of unique clicks recorded the delivery.
numClicks	long	The total number of clicks recorded the delivery.
avgClicks	double	The average number of clicks recorded the delivery.
uniqConversions	long	The number of unique conversions recorded for the delivery.
numConversions	long	The total number of conversions recorded for the delivery.
avgConversions	double	The number of average orders per contact recorded for the delivery.
numHardBadEmail	long	The bad email address bounce type means that the email server in question has indicated that this is not a valid account.


Name	Type	Description
numHardDestUnreach	long	The destination system unreachable bounce type means that there was a connection issue with the email server.
numHardMessageContent	long	The rejected due to message content hard bounce type means that the email server has identified the email as spam.
numHardBounces	long	The number of sent emails that were not delivered due to hard bounces.
numSoftBadEmail	long	The bad email address bounce type means that the email server in question has indicated that this is not a valid account.
numSoftDestUnreach	long	The destination system unreachable bounce type means that there was a connection issue with the email server.
numSoftMessageContent	long	Deferred due to message content is quite similar to the rejected message content bounce. Messages aren't often identified this way, as ISPs don't want you to resend something they identify as spam.
numSoftBounces	long	The number of sent emails that were not delivered due to soft bounces.
numOtherBounces	long	The number of times the delivery has bounced, where the bounces cannot be classified as hard or soft. We will try our best classify bounced deliveries as hard or soft, thus keeping them from showing up in this category.
numBounces	long	The total number of bounces recorded for the delivery.
numUnsubscribesByPrefs	long	The total number of contacts that were lost (i.e. made inactive) by unsubscribing.
numUnsubscribesByComplaint	long	The total number of contacts that were lost (made inactive) by complaining via an ISP feedback loop or the applications complaint system.
numViewsFacebook	long	The number of times the delivery was viewed after being shared using Facebook.


Name	Type	Description
numViewsTwitter	long	The number of times the delivery was viewed after being shared using Twitter.
numViewsLinkedIn	long	The number of times the delivery was viewed after being shared using LinkedIn.
numViewsDigg	long	The number of times the delivery was viewed after being shared using Digg.
numViewsMySpace	long	The number of times the delivery was viewed after being shared using MySpace.
numSocialViews	long	The number of times the delivery was viewed after being shared on social networks.

deliveryObject

A delivery contains the details on messages you have sent or scheduled to send, including associated content and statistics based on actions contacts took when they received the message associated with the delivery.

Name	Type	Description
id	string	The unique id assigned to the delivery.
start	dateTime	The date the delivery was scheduled to be sent.
messageId	string	The id of the message associated with the delivery.
status	string	The status of this delivery: <ul style="list-style-type: none"> • sent • sending • unsent • archived • skipped • tmp

Name	Type	Description
type	string	<p>The type of delivery:</p> <ul style="list-style-type: none"> • <code>normal</code> – A regular email delivery sent via the application GUI. • <code>test</code> – A test delivery. • <code>automated</code> – A delivery sent via an Automator or Automated Message Rule. • <code>split</code> – A delivery made as part of an A/B split test. • <code>transactional</code> – Transactional delivery • <code>triggered</code> – API triggered delivery. • <code>forwardtoafriend</code> – Forward to a friend delivery <p> Note: Only the <code>triggered</code>, <code>test</code>, and <code>transactional</code> types can be used with addDeliveries and updateDeliveries.</p>
fromEmail	string	The email address used in the From Address for this delivery.
fromName	string	The name used as the From Name for the delivery
replyEmail	string	The email address used as the Reply-To Address for the delivery.
authentication	boolean	<p>Enables sender authentication for the delivery if set to true. Sender authentication will sign your message with DomainKeys/DKIM (an authentication method that can optimize your message delivery to Hotmail, MSN, and Yahoo! email addresses). If you are associating this delivery with an automated message rule, these parameters will only be accepted if you clicked the Allow API to select sending options check box via the application UI on step 2 of creating an API triggered automated message rule.</p>

Name	Type	Description
replyTracking	boolean	Enables reply tracking for the delivery if set to true. Enabling Reply Tracking will store a copy of all replies to your messages on the Replies page. You may find this option convenient if you need someone other than the email address in the From line to read replies, or simply want the application to store replies. If you are associating this delivery with an automated message rule, these parameters will only be accepted if you clicked the Allow API to select sending options check box via the application UI on step 2 of creating an API triggered automated message rule.
messageRuleId	string	The ID of an automated message rule to associate with this delivery. Used to include this delivery in the reporting for the automator you specify.
optin	boolean	Whether or not the contact has opted in to receiving messages from you.
throttle	long	<p>Allows you to specify a throttle rate for the delivery. Throttle rate must be in range [0, 720] (minutes). For example you could specify 60 for the throttle range.</p> <p> Note: Throttling slows your email delivery speed, and thus spreads your email deliveries out over time. Many major ISPs track a sender's reputation based on the number of unsolicited email complaints that they generate over a certain period of time. The worse your sender reputation is with ISPs, the more likely you are to have a low sender rating in the application. Spreading your email delivery over time can help mitigate the impact of this issue, help ensure optimal deliverability, and mitigate traffic spikes on your website.</p>

Name	Type	Description
fatigueOverride	boolean	If set to <code>true</code> , the delivery can be sent even if it exceeds frequency cap settings for a contact.
content	messageContentObject[]	Read Only An array of the content (type, subject, content) used in the delivery.
recipients	deliveryRecipientObject[]	An array of the recipients who were, or are scheduled to receive the delivery. If the delivery was sent to a list, you will get back a list ID. If it was sent to a segment, you will get back a segment ID. If it was sent to an individual contact, you will get back a contact ID. If the delivery was sent to a combination of items (lists, segments, and contacts), you will get back an id for each item.
fields	messageFieldObject[]	An array of the API fields and data to substitute into the message being sent by this delivery.
products	deliveryProductObject[]	Specifies Product IDs to substitute for placeholders in product tags upon message send. Limit: 100 products
reemail	reemailObject[]	A reemail object used in the delivery. Remails allow you to send another email to contacts based on actions they did not take. The goal is to persuade them to continue along the conversion process.
numSends	long	The number of times this delivery has been sent.
numDeliveries	long	The number of successful deliveries.
numHardBadEmail	long	The bad email address bounce type means that the email server in question has indicated that this is not a valid account.
numHardDestUnreach	long	The destination system unreachable bounce type means that there was a connection issue with the email server.
numHardMessageContent	long	The rejected due to message content hard bounce type means that the email server has identified the email as spam.
numHardBounces	long	The number of sent emails that were not delivered due to hard bounces.

Name	Type	Description
numSoftBadEmail	long	The bad email address bounce type means that the email server in question has indicated that this is not a valid account.
numSoftDestUnreach	long	The destination system unreachable bounce type means that there was a connection issue with the email server.
numSoftMessageContent	long	Deferred due to message content is quite similar to the rejected message content bounce. Messages aren't often identified this way, as ISPs don't want you to resend something they identify as spam.
numSoftBounces	long	The number of sent emails that were not delivered due to soft bounces.
numOtherBounces	long	The number of times the delivery has bounced, where the bounces cannot be classified as hard or soft. We will try our best classify bounced deliveries as hard or soft, thus keeping them from showing up in this category.
numBounces	long	The total number of bounces recorded for the delivery.
uniqOpens	long	The number of unique opens recorded for the delivery.
numOpens	long	The total number of opens recorded for the delivery.
avgOpens	long	The number of average opens recorded for the delivery.
uniqClicks	long	The number of unique clicks recorded the delivery.
numClicks	long	The total number of clicks recorded the delivery.
avgClicks	double	The average number of clicks recorded the delivery.
uniqConversions	long	The number of unique conversions recorded for the delivery.
numConversions	long	The total number of conversions recorded for the delivery.
avgConversions	double	The number of average orders per contact recorded for the delivery.
revenue	decimal	The total revenue recorded for the delivery.

Name	Type	Description
numSurveyResponses	long	The number of survey responses generated via the delivery. This applicable if the delivery in question contained a survey.
numFriendForwards	long	The number of times a contact receiving the delivery forward it to a friend via our forward to a friend feature.
numContactUpdates	long	The number of times a contact updated their information using a Manage Preferences link contained in the delivery.
numUnsubscribesByPrefs	long	The total number of contacts that were lost (i.e. made inactive) by unsubscribing.
numUnsubscribesByComplaint	long	The total number of contacts that were lost (made inactive) by complaining via an ISP feedback loop or the applications complaint system.
numContactLoss	long	The contact loss metric represents the total number of contacts that were marked as inactive and can no longer receive marketing emails from you as a result of this delivery.
numContactLossBounces	long	The total number of contacts that were made inactive because they exceeded the bounce limit you have set in your account.
deliveryRate	double	The delivery rate recorded for the delivery. $(\text{Delivered}/\text{Sent}) * 100 = \text{Delivery Rate}$
openRate	double	The open rate recorded for the delivery. $(\text{Opens}/\text{Delivered}) * 100 = \text{Open Rate}$
clickRate	double	The click rate recorded for the delivery. $(\text{Clicks}/\text{Opens}) * 100 = \text{Click Rate}$
clickThroughRate	double	The click through rate recorded for the delivery. $(\text{Clicks}/\text{Delivered}) * 100 = \text{Click Through Rate}$
conversionRate	double	The conversion rate recorded for the delivery. $(\text{Conversions}/\text{Clicks}) * 100 = \text{Conversion Rate}$
bounceRate	double	The bounce rate recorded for the delivery.

Name	Type	Description
complaintRate	double	The complaint rate recorded for the delivery. (Complaints/Delivered) * 100 = Complaint Rate
contactLossRate	double	The contact loss rate recorded for the delivery. (Sent/Contact Loss) * 100 = Contact Loss Rate
numSocialShares	long	The total social shares recorded for the delivery.
sharesFacebook	long	The number of times the delivery was shared using Facebook.
sharesTwitter	long	The number of times the delivery was shared using Twitter.
sharesLinkedIn	long	The number of times the delivery was shared using LinkedIn.
sharesDigg	long	The number of times the delivery was shared using Digg.
sharesMySpace	long	The number of times the delivery was shared using MySpace.
numViewsFacebook	long	The number of times the delivery was viewed after being shared using Facebook.
numViewsTwitter	long	The number of times the delivery was viewed after being shared using Twitter.
numViewsLinkedIn	long	The number of times the delivery was viewed after being shared using LinkedIn.
numViewsDigg	long	The number of times the delivery was viewed after being shared using Digg.
numViewsMySpace	long	The number of times the delivery was viewed after being shared using MySpace.
numSocialViews	long	The number of times the delivery was viewed after being shared on social networks.
cartId	string	The ID of the shopping cart associated with the delivery.
orderId	string	The ID of the order associated with the delivery.
campaignId	long	The ID of the campaign associated with the delivery.

deliveryProductObject

Maps product placeholders used in product tags to Product IDs.

Name	Type	Description
placeholder	String	Placeholder text used in a product tag. Maximum of 50 characters.
productId	String	A valid Product ID that will replace the placeholder on message send. Maximum of 50 characters.

deliveryRecipientObject

Name	Type	Comments
deliveryType	string	Valid values are: <ul style="list-style-type: none"> • <code>eligible</code> - Indicates that the contact, list, keyword, or segment specified in this object are eligible to receive the message being sent to it. • <code>ineligible</code> - Indicates that the contact, list, keyword, or segment specified in this object are not eligible to receive the message being sent to it. • <code>selected</code> - (Default)
id	string	The unique id of the recipient receiving the delivery. Depending on the type specified, the id will be for a contact, a list, an smsKeyword, or a segment.
type	string	Whether the contact is receiving the message as part of a list, segment, SMS keyword, or as an individual contact. Valid values are: <ul style="list-style-type: none"> • <code>contact</code> • <code>list</code> • <code>segment</code> • <code>keyword</code> (SMS Only – Use with addSMSDeliveries)

deliveryRecipientStatObject

The `deliveryRecipientStatObject` contains information about deliveries made to a particular list, segment, or contact

Name	Type	Description
deliveryId	string	The unique id assigned to a delivery.
listId	string	The unique id assigned to the list the delivery was sent to.
segmentId	string	The unique id assigned to the segment the delivery was sent to.

Name	Type	Description
contactId	string	The unique id assigned to the contact the delivery was sent to. A contact id is only returned if the delivery was sent to a single contact.
numSends	long	The number of emails scheduled to be sent to the list, segment, or contact.
numDeliveries	long	The number of deliveries successfully delivered to the list, segment, or contact.
numHardBadEmail	long	The number of bad email address hard bounces.
numHardDestUnreach	long	The number of destination system unreachable hard bounces.
numHardMessageContent	long	The number of rejected due to message content hard bounces.
numHardBounces	long	The total number of hard bounces.
numSoftBadEmail	long	The number of bad email address soft bounces.
numSoftDestUnreach	long	The number of destination system temporarily unreachable soft bounces.
numSoftMessageContent	long	The number of deferred due to message content soft bounces.
numSoftBounces	long	The total number of soft bounces.
numOtherBounces	long	The number of bounces recorded that could not be classified as hard or soft.
numBounces	long	The total number of bounces recorded.
uniqOpens	long	The number of unique opens recorded.
numOpens	long	The total number of opens recorded.
avgOpens	double	The average number of opens recorded.
uniqClicks	long	The number of unique clicks recorded.
numClicks	long	The total number of clicks recorded.
avgClicks	double	The average number of clicks recorded.
uniqConversions	long	The number of unique conversions recorded.

Name	Type	Description
numConversions	long	The total number of conversions recorded.
avgConversions	double	The average number of conversions recorded.
revenue	long	The average number of conversions recorded.
numSurveyResponses	long	The total number of survey responses recorded.
numFriendForwards	long	The total number of friend forwards recorded.
numContactUpdates	long	The total number of contact updates recorded.
numUnsubscribesByPrefs	long	The total number of contacts who unsubscribed via a manage preferences webform.
numUnsubscribesByComplaint	long	The total number of contacts who were unsubscribed after complaining.
numContactLoss	long	The total number of contacts loss as a result of the delivery.
numContactLossBounces	long	The total number of contacts lost due to bounces.
deliveryRate	double	The delivery rate for the delivery.
openRate	double	The open rate for the delivery.
clickRate	double	The click rate for the delivery.
clickThroughRate	double	The click through rate for the delivery.
conversionRate	double	The conversion rate for the delivery.
bounceRate	double	The bounce rate for the delivery.
complaintRate	double	The complaint rate for the delivery.
contactLossRate	double	The contact loss rate for the delivery.

fieldObject

A field describes a single type of data which you can associate with your contacts. Imagine fields as extra columns of typed data that you can augment the contact's email address with. Examples might include information on your contacts' name and address, purchase history, demographics, and the like.

Overview

ou should not use fields to store particularly sensitive or private information about your contacts. Information such as credit card numbers, social security numbers, unencrypted passwords, and other similar data should be stored outside of the application in a system specifically designed for handling this type of data.

Any fielddata you use in a message must be smaller than 1 MB.

Name	Type	Description
id	string	The unique id of the field.
name	string	The internal name of the field.
label	string	The external (public facing) name of the field.

Name	Type	Description
type	string	<p>The type of field:</p> <ul style="list-style-type: none"> • <code>text</code> – Text box (max character limit: 65535) • <code>textarea</code> – A large multi-line text box (max character limit: 65535) • <code>password</code> – A text box that hides typed characters (max character limit: 65535) • <code>checkbox</code> – Checkboxes (max character limit: 1). Accepts a string value of <code>true</code> or <code>false</code>. • <code>radio</code> – Radio buttons. Accepts values that are predefined in the UI or a value that is passed in, if it is not already defined, but it will not be saved as a new option. • <code>select</code> – Pull-down menu • <code>integer</code> – Accepts any 10-digit, whole number less than or equal to 2147483647. If you need to store a static number (i.e. a number that is not incrementing or decrementing) we suggest using either a predefined field, or a custom field with a type of Text. For example, to store telephone numbers use one of the predefined fields created for storing telephone numbers. If you are storing incrementing or decrementing numbers in a custom field created with a type of Number, be mindful of the 10 digit, 2147483647 limit. • <code>currency</code> – Accepts any positive or negative number with two decimal points (max character limit: 15) • <code>float</code> – Accepts any positive or negative number with a decimal point (max character limit: 53) • <code>date</code> – A value that matches a specific date. See ../reference/r_api_soap_datevalue.xml for more information on the date format.

Name	Type	Description
visibility	string	The visibility selected for the field {public, private}. Public fields are visible to you and can be made visible to your contacts. Private fields are visible only to you.
options	fieldOptionObject[]	The possible options that can be set for a field if the field is a pull-down, check box, or radio button.

fieldOptionObject

Name	Type	Description
value	string	The value you want to specify for this particular field option. This only applies to pull-downs, check boxes, or radio buttons.
label	string	(Optional) Allows you to add a custom value for the field option. Custom values are used when you want the value (i.e. what is passed back to the server) for the option to be something other than what you added in for the value. If no label is specified, the value specified for the value property is used.
isDefault	boolean	If set to true, this field option will be used as the default.

headerFooterObject

Headers and Footers are reusable blocks of content which you can selective append to the top and bottom of your messages, respectively. This lets you create standard content to include things such as navigation areas, company contact information and unsubscribe links, etc.

Name	Type	Description
id	string	The unique id of assigned to the header/footer.
name	string	The name assigned to the header/footer.
html	string	The HTML version of the header.
text	string	The text version of the header
isHeader	boolean	Set to TRUE if the object is a header.

loginObject

The login object contains information about the user you are viewing, creating, or deleting. Information includes the username, password and permission information.

Name	Type	Description
username	string	The username assigned to the login.

Name	Type	Description
password	string	The password assigned to the login. We do not return the password when calling readLogins .
contactInformation	ContactInformation []	An array of the contact information set for the account. Click in the Type column for more information.
permissionAgencyAdmin	boolean	Gives the login agency administration permissions if you are creating a login for an agency account.
permissionAdmin	boolean	Gives the login administrative permission if you are creating a login for a client-account or a professional account.
permissionApi	boolean	Gives the login API permission if you are creating a login for a client-account or a professional account.
permissionUpgrade	boolean	Gives the login permission to purchase upgrades, such as inbox preview and additional fields. Applicable if you are creating a login for a client-account or professional account.
permissionFatigueOverride	boolean	Gives the login permission to override any contact frequency caps you have set for them. Applicable if you are creating a login for a client-account or professional account.
permissionMessageCompose	boolean	Gives the login permission to create messages if you are creating a login for a client-account or a professional account.
permissionMessageApprove	boolean	Gives the login permission to approve messages for sending.
permissionMessageDelete	boolean	Gives the login permission to delete messages if you are creating a login for a client-account or a professional account.
permissionAutomatorCompose	boolean	Gives the login permission to create automated message rules if you are creating a login for a client-account or a professional account.
permissionListCreateSend	boolean	Gives the login permission to create, and send to lists if you are creating a login for a client-account or a professional account.

Name	Type	Description
permissionListCreate	boolean	Gives the login permission to create, but not send messages to lists if you are creating a login for a client-account or a professional account.
permissionSegmentCreate	boolean	Gives the login permission to create segments if you are creating a login for a client-account or a professional account.
permissionFieldCreate	boolean	Gives the login permission to create fields if you are creating a login for a client-account or a professional account.
permissionFieldReorder	boolean	Gives the login permission to create messages if you are creating a login for a client-account or a professional account.
permissionSubscriberCreate	boolean	Gives the login permission to create contacts if you are creating a login for a client-account or a professional account.
permissionSubscriberView	boolean	Gives the login permission to view contacts if you are creating a login for a client-account or a professional account.

mailListObject

A list is a logical collection of contacts in your account. Lists can be managed either entirely internally via your interactive use of the application, with API calls, or you can place them on Webforms and allow contacts to add or remove themselves from lists. You can use lists to collect contacts with similar interests or profiles. Lists, along with Segments and individual contacts, can be used as sending targets for a delivery.

Name	Type	Description
id	string	The unique id assigned to the list. You can obtain the id for a list by calling readLists , or by looking at the footer when viewing the overview page for an individual list in the application.
name	string	The internal name of the list.
label	string	The external (customer facing) name of the list.
activeCount	long	The number of active contacts of currently on the list.
status	string	The status of the list. Valid values are active, deleted, and tmp

messageContentObject

A single message can have two different content objects associated with it for text and HTML versions of the message. A contact's message format preference will indicate which type of content they should get when receiving a delivery.

Name	Type	Description
type	string	The type of message content {html or text}.
subject	string	The subject line used in the the message.
content	string	The actual content of the message.

messageFieldObject

Any fielddata you use in a message must be smaller than 1 MB.

Name	Type	Required	Comments
name	string	Yes	The name of the API message tag. API message tags can be placed in the body or subject line of an email message with a prepended '#', as in "%#tag_name%"). When you reference the name of an API message tag via the API, be sure to leave off the "%# %" portion of the API message tag. For example, name => tag_name, rather than name => %#tag_name%. Loop tags use a slightly different syntax (%#tagname_#%) and can only be added to the body of an email message. When you reference the the name of a Loop tag via the API, be sure to leave off the "%# %" portion of the Loop tag, and replace the underscore "_" with an underscore followed by a number. For an example of how to use Loop tags, see the code example on the addDeliveries page.
type	string	Yes	The version of the message into which API message tag content should be inserted {text or html}.

Name	Type	Required	Comments
content	string	Yes	The value being inserted into the body of the message via the API message tag. Links included the content of a messageFieldObject are Dynamic Links as opposed to Static links which are included in the message content. When you insert dynamic links in messages via the API, Bronto will only track up to 30 dynamic links for single send messages and up to 10 for bulk messages. If it's important to track all dynamic links, add links directly to the design of the message in Bronto.

messageFolderObject

Message folders provide you with a way to logically organize and group your messages. Message folders can be nested inside other message folders for further organization.

Name	Type	Description
id	string	The unique id assigned to the folder.
name	string	The name assigned to the folder.
parentId	string	The unique id assigned to the parent folder which contains this folder.
parentName	string	The name assigned to the parent folder which contains this folder.

messageObject

A message provides a way to store email content with Bronto. Along with the content, you can give the message a descriptive name and optionally place it inside a message folder for organization purposes. The actual content is stored inside messageContentObject(s).

Name	Type	Description
id	string	The unique id assigned to the message. You can obtain the id for a message by calling readMessages , or by looking at the footer when viewing the overview page for an individual message in the application.
name	string	The name assigned to the message

Name	Type	Description
status	string	Read Only The status of the message. Valid values are: <ul style="list-style-type: none"> active delete tmp
messageFolderId	string	The unique id of the folder containing the message.
campaignId	long	The ID of the campaign associated with the message.
content	messageContentObject[]	An array of content objects {type, subject, content} for the message.

messageRuleObject





Message rules provide ways to automatically trigger message deliveries based on a matching set of conditions. They also allow you to interactively view aggregate reporting statistics for all associated deliveries inside the Bronto application.


Name	Type	Description
id	string	The unique id assigned to the Automated Message Rule.
name	string	The name assigned to the Automated Message Rule.
type	string	The type assigned to the Automated Message Rule {activity-based, date-based, schedule recurring, and api triggered.}
messagedId	string	The unique id assigned to the message that will be sent via the Automated Message Rule.

orderObject

The order object represents order data you can pass into the application. Any field data you use in a message must be smaller than 1 MB.

Name	Type	Description
id	string	The unique id for the order. If the id for the order already exists within the application, then this API call will update that order. However, if the id for the order does not exist within the application then this API call will create a new order.

Name	Type	Description
contactId	string	<p>A unique id assigned to the contact placing the order.</p> <p> Note: If the order id already exists, then this field is ignored and not updated</p>
email	string	<p>The email address of the person placing the order.</p> <p> Note: If a corresponding contact does not exist, then a new one will be created with the status of <code>transactional</code>. If the order id already exists, then this field is ignored and not updated</p>
products	<code>productObject[]</code>	<p>Array of the products contained in this order.</p> <p> Note: This is not required. You can add an order and come back later to fill in the product details. When the order id does already exist:</p> <ul style="list-style-type: none"> • If this is not provided at all (as in null) then it is not replaced • If an empty array is provided (as in not null but the array has no values), then all products are removed from the order • If an array with values is provided, then all products in the order are replaced with this new set
orderDate	dateTime	<p>Date and time of the order.</p> <p> Note: If no value is provided, the system will timestamp the record. You can (and should) specify a timezone offset if you do not want the system to assume you are providing a time in UTC (Coordinated Universal Time / Greenwich Mean Time). For the Eastern Time Zone on Daylight Savings Time, this would be:</p> <p><i>YYYY-MM-DDTHH:MM:SS-04:00</i></p>

Name	Type	Description
deliveryId	string	The id of the delivery the order is associated with.
messageId	string	The id of the message the order is associated with.
automatorId	string	The id of the automator the order is associated with.
listId	string	The id of the list the order is associated with.
segmentId	string	The id of the segment the order is associated with.
deliveryType	string	<p>The type of delivery the order resulted from. Valid values are:</p> <ul style="list-style-type: none"> • normal • test • transactional • automated <p>If you are on the old order service and you try to pass a type that is not one of the 4 valid types then type will be ignored.</p>
tid	string	<p>Unique id that associates an order with a specific contact and delivery. Since this ties the order back to a delivery, it is then classified as a conversion for that specific delivery.</p> <p> Note: The <code>tid</code> can only be pulled from the Bronto tracking cookie, which requires that Conversion Tracking be enabled within the Bronto application. If the order <code>id</code> already exists, then this field is ignored and not updated</p>

productObject

The product object represents data about a product you can pass into the application as part of the orderObject. Any field data you use in a message must be smaller than 1 MB.

Name	Type	Description
sku	string	The unique id, or SKU, of the product.
name	string	The name of the product
description	string	The description of the product.
category	string	The category for the product

Name	Type	Description
image	string	URL that points to an image of the product.
url	string	URL that points to the web page of the product.
quantity	int	Number of units of the product that are included in this order.
price	decimal	Unit price of the product.

recentActivityObject

The `recentActivityObject` contains activity data for an account.

Name	Type	Description
createdDate	dateTime	The date the activity was recorded.
contactId	string	The ID assigned to the contact associated with the activity.
listId	string	The ID assigned to the list that the delivery associated with the activity was sent to.
segmentId	string	The ID assigned to the segment that the delivery associated with the activity was sent to.
keywordId	string	The ID assigned to the SMS keyword that the SMS delivery associated with the activity was sent to.
messageId	string	The ID assigned to the message associated with the activity.
deliveryId	string	The ID assigned to the delivery associated with the activity.
workflowId	string	The ID assigned to the workflow that sent the delivery associated with the activity.

Name	Type	Description
activityType	string	<p>The type of activity the object represents. The activityType can be:</p> <p>Inbound Activities:</p> <ul style="list-style-type: none"> • bounce • click • contactSkip • conversion • friendforward • open • reply • sms_reply • sms_bounce • social • unsubscribe • webform <p>Outbound Activities:</p> <ul style="list-style-type: none"> • send • sms_send
emailAddress	string	<p>The email address of the contact associated with the activity. The emailAddress li is returned if a contactId is returned, and an email address is stored for the associated contact.</p>
mobileNumber	string	<p>The mobile number of the contact associated with the activity. The mobileNumber li is returned if a contactId is returned, and a mobile number is stored for the associated contact.</p>

Name	Type	Description
contactStatus	string	<p>The status of the contact associated with the activity. Each valid status is described below:</p> <ul style="list-style-type: none"> • active – Active contacts are “live” contacts that can receive both marketing and transactional emails from you. • onboarding – Onboarding contacts are new contacts who are waiting to be sent their first marketing email from you. Once they are sent the first marketing email, they will enter the assessment phase of the automated onboarding process and automatically move from onboarding to Awaiting Assessment, and then to active. You can not update a contact’s status if they currently have a status of onboarding. Contacts with a status of onboarding can receive transactional emails from you. • transactional – Contacts with a status of transactional can only be sent transactional emails. A transactional email facilitates an agreed-upon transaction, or updates a customer in an existing business relationship. • bounce – The bounce contact status indicates that you cannot send messages to this contact because they have received a hard bounce due to bad email address, or they have exceed the bounce limit for your account. • unconfirmed – Contacts with a status of unconfirmed have not yet agreed to receive your marketing email messages. Remember, one of our policies is that contacts must agree to receive your marketing email messages in order for you to send marketing emails to them. Contacts with a status of unconfirmed can receive transactional emails from you. • unsub – Contacts with a status of unsub have either unsubscribed themselves from receiving marketing email messages from you, or were unsubscribed by you. Contacts with a status of unsub can receive transactional emails from you.

Name	Type	Description
messageName	string	The name of the message associated with the activity. The messageName li is returned if a messageId is returned
deliveryType	string	<p>The type of delivery associated with the activity:</p> <ul style="list-style-type: none"> • bulk – A regular email delivery sent via the application GUI. • test – A test delivery. • automator – A delivery sent via an Automator or Automated Message Rule. • split – A delivery made as part of an A/B split test. • transaction – Transactional delivery • trigger – API triggered delivery. • ftaf – Forward To A Friend delivery. <p>The deliveryType li is returned if a deliveryId is returned.</p>
deliveryStart	dateTime	The date/time the delivery associated with the activity was scheduled. The deliveryStart li is returned if a deliveryId is returned.
workflowName	string	The name of the workflow associated with the activity. The workflowName li is returned if a workflowId is returned.
segmentName	string	The name of the segment associated with the activity. The segmentName li is returned if a segmentId is returned.
listName	string	The name of the list associated with the activity. The listName li is returned if a listId is returned.
listLabel	string	The label assigned to the list associated with the activity. The label is the external (customer facing) name given to a list. The listLabel li is returned if a listId is returned.
automatorName	string	The name of the automator associated with the activity.


Name	Type	Description
smsKeywordName	string	The name of the SMS keyword associated with the activity. The smsKeywordName li is returned if a keywordId is returned.
bounceType	string	<p>The type of bounce recorded. The following types can be returned:</p> <p>Hard Bounces</p> <ul style="list-style-type: none"> • bad_email • destination_unreachable • rejected_message_content <p>Soft Bounces</p> <ul style="list-style-type: none"> • temporary_contact_issue • destination_temporarily_unavailable • deferred_message_content • unclassified <p>The bounceType li is returned if the activityType is bounce.</p>
bounceReason	string	The detailed reason why the bounce occurred. The bounceReason li is returned if the activityType is bounce.
skipReason	string	The detailed reason why the contact was skipped when attempting to send to them. The skipReason li is returned if the activityType is contactSkip.
linkName	string	The name of the link that was clicked. The linkName li is returned if the activityType is click.
linkUrl	string	The URL of the link that was clicked. The linkUrl li is returned if the activityType is click.
orderId	string	The ID assigned to the order. The orderId li is returned if the activityType is conversion.

Name	Type	Description
unsubscribeMethod	string	<p>The method used by the contact to unsubscribe. Valid values are:</p> <ul style="list-style-type: none"> • subscriber • admin • bulk • listcleaning • fbl (Feedback Loop) • complaint • account • api • unclassified <p>The unsubscribeMethod li is returned if the activityType is unsubscribe.</p>
ftafEmails	string	<p>The emails that were used in the Forward To A Friend Delivery. The ftafEmails li is returned if the activityType is friendforward.</p>
socialNetwork	string	<p>The social network the activity was performed on. The valid networks are:</p> <ul style="list-style-type: none"> • facebook • twitter • linkedin • digg • myspace <p>The bounceType li is returned if the activityType is social.</p>
socialActivity	string	<p>The activity performed. The valid activities are:</p> <ul style="list-style-type: none"> • view • share <p>The socialActivity li is returned if the activityType is social.</p>
webformId	string	The unique ID for a webform.

Name	Type	Description
webformAction	string	The activity performed on the webform. Valid values are: <ul style="list-style-type: none"> submitted view The webformAction is returned if the activityType is webform.
webformName	string	The name of the webform used. The webformName is returned if the activityType is webform.

remailObject

The remail object contains data about a remail. Remails allow you to send another email to contacts based on actions they did not take. The goal is to persuade them to continue along the conversion process. The remail object is used with the addDeliveries call.

Name	Type	Description
days	int	The number of days until the remail will trigger.
time	string	The time at which the remail will trigger. The time should be in 24 hour format: HH:MM:SS  Note: Do not add a time zone offset for the remail time. The time for the remail will use the time zone specified for the delivery object.
subject	string	The subject line to be used for the remail.
messageId	string	The unique id assigned to the message being used for the remail.
activity	string	The activity that will trigger the remail. Valid values are: <ul style="list-style-type: none"> noopen opennoclick clicknoconvert

segmentCriteriaObject

The segmentCriteriaObject has been deprecated. While it is possible, but unlikely, that your call might return this data, it is not reliable.

Name	Type	Valid Values	Description
operator	string	<ul style="list-style-type: none"> • email – Segment based on an email address • source – Segment based on how a contact was added. • message_preference – Segment based on a contact's message preference. • subscriber_ctime – Segment based on the date a contact was created. • subscriber_mtime – Segment based on the date a contact's data was last modified or updated. • subscriber_bounced – Segment based on the last time a delivery sent to a contact bounced. • field – Segment based on field data • list – Segment based on list data • sent – Segment based on whether a particular delivery was sent. • sentbounce – Segment based on deliveries that were sent, but bounced • open – Segment based on opens. • conv – Segment based on conversions. • click – Segment based on clicks. 	The operator used to define the rule.

Name	Type	Valid Values	Description
condition	string	<ul style="list-style-type: none"> • = • != • < • > • <= • >= • within_last • within_next • start • not_start • end • not_end • in • not_in • option • not_option • same_month_day • empty • not_empty 	The condition used to refine the rule.
value	string	The value used in the rule.	

segmentObject

A segment is a logical, dynamic group of contacts that match a certain set of criteria that you define. Think of segments as dynamic lists. Criteria can include matching based on contact field data, contact behavior (opening/clicking messages, etc.), and more. You can use segments to automatically group together contacts for reporting and sending purposes that match conditions important to your business needs.


Name	Type	Description
id	string	The unique id assigned to the segment. You can obtain the id for a segment by calling readSegments , or by looking at the footer when viewing the overview page for an individual segment in the application.
name	string	The name assigned to the segment.
rules	segmentRuleObject[]	The segmentRuleObject has been deprecated. While it is possible, but unlikely, that your call might return rules this data is not reliable. The rules used to defined which contacts will belong to the segment.
lastUpdated	dateTime	The date the segment was last updated or ran.
activeCount	long	The total number of active contacts currently belonging to the segment.

segmentRuleObject

The segmentRuleObject has been deprecated. While it is possible, but unlikely, that your call might return rules this data is not reliable.

Name	Type	Description
canMatchAny	boolean	Determines if contacts on the segment can match any of the rules defining the segment (TRUE), or if the contacts have to match every rule that defines the segment (FALSE).
criteria	segmentCriteriaObject	The segmentCriteriaObject has been deprecated. While it is possible, but unlikely, that your call might return this data, it is not reliable. The criteria used to define a rule used in the segment.


smsDeliveryContactsObject

Name	Type	Required	Comments
keyword	string	Yes	The id for the keyword you want to send to. If you only want to send to specific contacts subscribed to the keyword, you can pass in their ids using the <code>contactIds</code> attribute below.  Note: If no <code>contactIds</code> are specified, then all contacts subscribed to the keyword will receive the SMS message.
contactIds	string	No	The id(s) of the contacts on the keyword you want to send to. This can be a single <code>contactId</code> or an array of <code>contactIds</code> .

smsDeliveryObject

The smsDeliveryObject contains data for an SMS delivery. smsDeliveryObjects contain details on SMS deliveries you have sent or have scheduled to send.

Name	Type	Description
id	string	The unique id assigned to the SMS delivery.
start	dateTime	The date the delivery was/is scheduled to be sent.
messageId	string	The id associated with the SMS message used in the SMS delivery.
deliveryType	string	The deliveryType indicates the how the SMS delivery was made. Valid values are: <ul style="list-style-type: none"> • bulk • test • workflow • transaction

Name	Type	Description
status	string	The status of the SMS delivery. Valid values are: <ul style="list-style-type: none"> • unsent • sending • sent • skipped • archived
content	string	The content used in the SMS delivery. <p> Note: SMS messages are limited to 160 characters. The following text must be included somewhere in the body of the SMS message: Text STOP to end</p>
recipients	deliveryRecipientObject[]	An array of the recipients who were, or are scheduled to receive the SMS delivery.
fields	smsMessageFieldObject[]	An array of the API fields and data to substitute into the SMS message being sent by this SMS delivery.
numSends	long	The total number of text messages sent in the SMS delivery.
numDeliveries	long	The total number of successful text messages sent in the SMS delivery.
numIncoming	long	The total number of replies to the SMS delivery.
numBounces	long	The total number of bounces recorded for the SMS delivery.
deliveryRate	double	The percentage of text messages that were successfully delivered in the SMS delivery.

smsKeywordObject

The smsKeywordObject is used to add, update, and delete SMS keywords via the API. You can also add contacts to, and remove contacts from specific SMS keywords.

Name	Type	Description
id	string	The unique id for the SMS keyword.
name	string	The name assigned to the SMS keyword.
description	string	The description provided for the SMS keyword.
subscriberCount	long	The number of people subscribed to the SMS keyword.

Name	Type	Description
frequencyCap	long	The frequency cap represents the maximum number of SMS messages you can send to a person subscribed to the keyword each month. Best practice suggests you set the frequency cap to 30 or less per month. Valid values are 1-30.
dateCreated	dateTime	The date and time the SMS keyword was created.
scheduledDeleteDate	dateTime	When you delete an SMS keyword, it is marked for deletion and then deleted 7 days later. The scheduledDeleteDate represents the date the SMS keyword will be deleted.
confirmationMessage	string	The confirmation message set for the SMS keyword. The confirmation message is sent when a user confirms their subscription to the SMS keyword. The text “ <i>Txt STOP to <XXXXX> to end, HELP for info. <XX>msg/mo. Msg&Data rate may apply.</i> ” will be appended to your message. The confirmationMessage can be a maximum of 83 characters.
messageContent	string	The message content set for the SMS keyword. The messageContent will be sent each time a contact texts into the keyword or replies to a message from the keyword. The messageContent can be a maximum of 160 characters.



Name	Type	Description
keywordType	string	<p>The type set for the SMS keyword. Valid values are:</p> <ul style="list-style-type: none"> • <code>basic</code> – Basic keywords are non-subscription keywords meant for individual transactions. With basic keywords, a person texts into a keyword and a response is sent back. The interaction ends there. The recipient is not added to a list because they have not agreed to receive future marketing messages from you. • <code>subscription</code> – Subscription based keywords require a person to choose to receive SMS messages from you by texting into a given keyword. Contacts who subscribe to a subscription based keyword will be added to a list so that you can send SMS messages to them in the future. • <code>text2join</code> – <code>text2join</code> keywords allows you to grow your list by providing a way for potential contacts to text their email address in and automatically be added to one of your lists.

smsMessageFieldObject

Name	Type	Required	Comments
name	string	Yes	The name of the API message tag used in the SMS message. API message tags can be placed in the body or subject line of an email message with a prepended '#', as in "%#tag_name%". When you reference the name of an API message tag via the API, be sure to leave off the "%# %" portion of the API message tag.
content	string	Yes	The value being inserted into the body of the SMS message via the API message tag.

smsMessageObject

The `smsMessageObject` contains data for an SMS message. Along with the content, you can give the SMS message a descriptive name and optionally place it inside a message folder for organization purposes.

Name	Type	Description
<code>id</code>	string	The unique id assigned to the SMS message.
<code>name</code>	string	The name assigned to the SMS message.
<code>status</code>	string	Read Only The status of the SMS message. Valid values are: <ul style="list-style-type: none"> • <code>active</code> • <code>deleted</code> • <code>tmp</code>
<code>messageFolderId</code>	string	The unique id of the folder containing the SMS message.
<code>shortenUrls</code>	boolean	Indicates if the SMS message uses shortened URLs.  Note: In addition to shortening URLs, which allows for more content, this setting also allows the application to track the URLs used in the message. URLs will be shortened when the message is sent. The character count takes this into account, and thus represents the count for the message as if the URLs were shortened.
<code>content</code>	string	The content of the SMS message.  Note: SMS messages are limited to 160 characters. The following text must be included somewhere in the body of the SMS message: Text STOP to end

unsubscribeObject

The `unsubscribeObject` contains data about unsubscribes. A contact can unsubscribe by you, or they can unsubscribe themselves via an Unsubscribe Webform or a Manage Preferences Webform.

Name	Type	Description
<code>contactId</code>	string	The unique ID of the contact associated with the unsubscribe.

Name	Type	Description
deliveryId	string	The unique ID of the delivery that resulted in the contact unsubscribing.
method	string	The method used by the contact to unsubscribe. The valid methods are: <ul style="list-style-type: none"> • subscriber • admin • bulk • listcleaning • fbl (Feedback loop) • complaint • account • api
complaint	string	Optional additional information about the unsubscribe.
created	dateTime	The date/time the unsubscribe was created.

webformObject

The webform object contains data about webforms.

Name	Type	Description
id	string	The unique ID for the webform.
name	string	The name assigned to the webform.
type	string	The type assigned to the webform. Valid values are: <ul style="list-style-type: none"> • managePreferences • addContact • forwardToAFriend • lookupContact • thankYou • confirmation • unsubscribe • complaint
isDefault	boolean	Set to true if the webform is a default.
modified	dateTime	The last time the webform was modified.

workflowObject

The workflow object contains information about workflows in your account.

Name	Type	Description
id	string	The unique id assigned to the workflow.

Name	Type	Description
siteId	string	The siteId associated with the workflow.
name	string	The name assigned to the workflow
description	string	The description added for the workflow.
status	string	The status of the workflow. A workflow can have a status of active or inactive . Active workflows are live workflows that are running. Inactive workflows are not running and do not perform any actions.
createdDate	dateTime	The date the workflow was created.
modifiedDate	dateTime	The last time the workflow was modified.
activatedDate	dateTime	The last time the workflow was activated.
deActivatedDate	dateTime	The last time the workflow was deactivated (made inactive).

SOAP Data Formats

Data formats are standard and are defined by the SOAP specification.

Date Formats

SOAP uses yyyy-mm-ddThh:mm:ss.nnn+|-hh:mm as the date/time format. For example, December 14 1984 12:14:37 would be 1984-12-14T12:14:37.000-04:00. You can (and should) specify a timezone offset if you do not want the system to assume you are providing a time in UTC (Universal Coordinated Time / Greenwich Mean Time). For the Eastern Time Zone on Daylight Savings Time, this would be YYYY-MM-DDThh:mm:ss.SSS-04:00

REST API

Bronto's newest offerings leverage the convenience and flexibility of REST. With our REST API client, you can access and work with your product and order data.

Our REST API library is categorized by service. If you are not working with products, orders, or carts you will need to reference the SOAP API library to find the information you need to work with other objects in Bronto.

Product Service

The Product Service helps you to manipulate product data in your product catalog and to import product data feeds. It is accessible using HTTPS and secured with OAuth 2.0. The code samples in the Product Service REST API library are written in Python using the Requests library.

Order Service

The Order Service helps you to view, add, update, and delete order and cart data. You can also use order service APIs to manipulate the status of a cart. The code samples in the Order Service REST API library are written in Java.

Getting Started With REST

Bronto's newest offerings, our product and order services, maximize the flexibility of the REST web service.

All API calls require the existence of an active, authenticated session. Therefore, before you can use our API reference library to interact with Bronto's API you will need to enable API access using the platform and set up authentication that you can use to log in.

In addition to REST-specific access and configuration instructions, we've provided additional topics designed to help you get up and running.

REST Authentication

Bronto's REST API is accessible using HTTPS and secured with OAuth 2.0. In order to setup the API integration you will need to configure and use the Hallmonitor Client.

Each request to the external API must include an HTTP Authorization header. The header value should specify a "Bearer" authorization scheme followed by a space and then the OAuth2 access token.

A 403 Forbidden response will be returned if no authorization header is provided, if a scheme other than Bearer is specified, if the token is unknown/expired, or if the token has an insufficient scope to execute the request.

Making REST Calls

REST allows you to make simple HTTP requests using calls from a client to a server in order to work with your Bronto data.

The base URI for our REST API is `https://rest.bronto.com/`. Our API supports the following Methods:

- GET: Used to view or read data already stored in Bronto. If you plan on updating data, particularly in the Order Service, it is a good idea to GET the data first in order to ensure your updates won't have unintended consequences.
- POST: Primarily used to add new content, though some Order Service calls also use POST to manipulate existing content.
- PUT: Used to make changes to existing content.
- DELETE : Used to remove content from Bronto. Because Bronto's data is used in so many different scenarios, it is important to always give careful consideration to the impact of a DELETE before performing it.

You can use any language to interact with our REST API. We provide code samples for each call in PHP and Java and our Product Service also provides Python examples.

REST Authentication

Bronto's REST API is accessible using HTTPS and secured with OAuth 2.0. In order to setup the API integration you will need to configure and use the Hallmonitor Client.

Each request to the external API must include an HTTP Authorization header. The header value should specify a "Bearer" authorization scheme followed by a space and then the OAuth2 access token.

A 403 Forbidden response will be returned if no authorization header is provided, if a scheme other than Bearer is specified, if the token is unknown/expired, or if the token has an insufficient scope to execute the request.

REST Order Service

The Order Service provide REST APIs for interacting with carts and orders. Common calls for orders include Get, Add, Update, and Delete. Carts include all of these calls and Fiddle, Abandon, Expire, and Delete.

If you are not using Order Service for order data, you should use the SOAP functions `addOrUpdateOrders` or `deleteOrders` to work with order data instead of the REST API.

Add Order

Allows you to add a new order.

Overview

This request accepts both representations of an order's state via the "states" and "status" fields. Updates to either field will be reflected in the other. The "status" field is currently deprecated and it is recommended that you use the "states" field going forward. If both "status" and "states" are both specified, the data supplied in "status" will be used.

This request requires an access token with "orders/carts-write" scope. If the request is successful, a 201 Created response will be returned with the newly created Order in the response body. If an Order or Return with the same customerOrderId already exists, a 409 Conflict response will be returned. If there is a problem with any of the data in the request body, a 400 Bad Request response will be returned with an explanation of the error. If the orderDate is earlier than January 1st, 2000 or more than 5 years in the future, then the order request is rejected with a 400 Bad Request response.

You can provide tracking attribution by including a tid with your request or a TrackingCookieName and TrackingCookieValue pair. For more information see Orders REST API Tracking Attribution.

URI

```
POST: https://rest.bronto.com/orders?
createContact={boolean}&triggerEvent={boolean}&force={boolean}&ignoreInvalidTid={boolean}
```

Parameters

Parameter	Type	Description
createContact	Optional Boolean	Determines behavior when the request contains an email address that is not associated with a Contact. If true, a transactional Contact will be created. If false, the service will return a 409 Conflict response. The default value is false.
triggerEvents	Optional Boolean	Determines whether workflows or other contact interactions can be triggered by the state of this order. If false, workflow nodes do not fire. The default value is true.
force	Optional Boolean	Determines whether workflows or other contact interactions will be triggered by the state of this order, regardless of whether the state has changed. If true, an order that is currently processed will re-trigger, potentially re-triggering a workflow that might message the customer. If false, workflows and other contact interactions are only triggered when the state changes, such as a pending order changing to processed.
ignoreInvalidTid	Optional Boolean	Determines behavior when the request contains an invalid TID or TrackingCookieName and TrackingCookieValue pair. If true, the service will ignore the invalid values and continue.

Request Body

The Order data used to create the Order.

```
{
  status:PENDING | PROCESSED
  discountAmount:number
  emailAddress:validly formatted email address
  contactId:string
  grandTotal:number
  deliveryId:string
  lineItems: [
    {
      name:string
      description:string
      sku:string
      other:string
      imageUrl:string
      category:string
      productUrl:string
      quantity:number
      salePrice:number
      totalPrice:number
      unitPrice:number
    }
  ]
  originIp: string
  messageId:string
  originUserAgent: string
  shippingAmount: number
  shippingDate: ISO-8601 datetime
  shippingDetails: string
  shippingTrackingUrl: string
  subtotal: number
  taxAmount: number
  trackingCookieName: string
  trackingCookieValue: string
  tid:string
  cartId:UUID
  customerOrderId:string
  orderDate:ISO-8601 datetime
  currency:ISO-4217 currency code
  states: {
    processed:boolean
    shipped:boolean
  }
}
```

Response Body

A response containing the newly created order.

```
{
  emailAddress:validly formatted email address
  contactId:string
  orderDate:ISO-8601 datetime
  status:PENDING | PROCESSED
  hasTracking:boolean
  trackingCookieName:string
  trackingCookieValue:string
  deliveryId:string
  customerOrderId:string
  discountAmount:number
}
```

```

grandTotal:number
lineItems:[
  {
    name:string
    other:string
    sku:string
    category:string
    imageUrl:string
    productUrl:string
    quantity:number
    salePrice:number
    totalPrice:number
    unitPrice:number
    description:string
    position:number
  }
]
messageId:string
originIp:IPv4 or IPv6 address
originUserAgent:string
shippingAmount:number
shippingDate:ISO-8601 datetime
shippingDetails:string
shippingTrackingUrl:string
subtotal:number
taxAmount:number
cartId:UUID
createdDate:ISO-8601 datetime
updatedDate:ISO-8601 datetime
currency:ISO-4217 currency code
states: {
  processed:boolean
  shipped:boolean
}
orderId:UUID
}

```

Java Code Example

```

import java.util.Arrays;
import java.util.HashMap;
import java.util.Map;
import java.util.UUID;

import javax.ws.rs.client.Client;
import javax.ws.rs.client.ClientBuilder;
import javax.ws.rs.client.Entity;
import javax.ws.rs.core.Form;
import javax.ws.rs.core.MediaType;
import javax.ws.rs.core.Response;

public class AddOrderExample {

    // Host
    private static final String BRONTO_HOST = "http://rest.bronto.com";
    private static final String BRONTO_AUTH_PATH = "https://auth.bronto.com/
oauth2/token";

    // Paths
    private static final String ADD_ORDER_PATH = "orders";
    private static final String CREATE_CONTACT = "createContact";
}

```

```

private static final String IGNORE_INVALID_TRACKING
= "ignoreInvalidTracking";

// OAuth Request property names
private static final String GRANT_TYPE = "grant_type";
private static final String CLIENT_ID = "client_id";
private static final String CLIENT_SECRET = "client_secret";

// OAuth Request property values
private static final String CLIENT_CREDENTIALS = "client_credentials";
private static final String EXAMPLE_CLIENT_ID = "XXXXXXXXXXXXXXXXXXXXXXXXXXXX";
private static final String EXAMPLE_CLIENT_SECRET
= "XXXXXXXXXXXXXXXXXXXXXXXXXXXX";

private static final String ACCESS_TOKEN = "access_token";

private static final String REASON_HEADER = "X-Reason";

public static void main(String[] args) {
    Client client = ClientBuilder.newClient();

    // To be able to access Orders Rest API, you need an access token.
    // First, we build the request data needed to gain an access token
    Form requestData = new Form();
    requestData.param(GRANT_TYPE, CLIENT_CREDENTIALS);
    requestData.param(CLIENT_ID, EXAMPLE_CLIENT_ID);
    requestData.param(CLIENT_SECRET, EXAMPLE_CLIENT_SECRET);

    // Then build and send the request
    Response oauthResponse = client.target(BRANTO_AUTH_PATH)
        .request(MediaType.APPLICATION_JSON)
        .accept(MediaType.TEXT_PLAIN_TYPE)
        .post(Entity.form(requestData));

    if (oauthResponse.getStatus() != Response.Status.OK.getStatusCode()) {
        throw new RuntimeException("Unable to get access token.");
    }

    // Retrieve the access token from the response
    Map<String, Object> responseData =
    oauthResponse.readEntity(Map.class);
    UUID accessToken = UUID.fromString((String)
    responseData.get(ACCESS_TOKEN));

    // Now to add an Order, we first build the request data
    Map<String, Object> lineItemData = new HashMap<String,
    Object>(2);
    lineItemData.put("name", "Coffee Mug");
    lineItemData.put("description", "A cool coffee mug");
    lineItemData.put("category", "Kitchen Stuffs");
    lineItemData.put("other", "5oz");
    lineItemData.put("imageUrl", "http://www.example.com/
    images/0003105.jpg");
    lineItemData.put("productUrl", "http://www.example.com/
    products/0003105");
    lineItemData.put("sku", "0003105");
    lineItemData.put("quantity", 1);
    lineItemData.put("totalPrice", 3.99f);
    lineItemData.put("unitPrice", 3.99f);
    lineItemData.put("salePrice", 3.99f);

```

```

Map<String, Object> orderData = new HashMap<String, Object>
(2);
orderData.put("customerOrderId", "abc-123");
orderData.put("cartId", "7255c31c-39ac-4611-a2b7-974b0ebfa555");
orderData.put("emailAddress", "example@email.com");
orderData.put("currency", "USD");
orderData.put("grandTotal", 4.27f);
orderData.put("discountAmount", 0);
orderData.put("taxAmount", 0.28f);
orderData.put("subtotal", 3.99f);
orderData.put("lineItems", Arrays.asList(lineItemData));
orderData.put("orderDate", "2015-01-01");
orderData.put("status", "PROCESSED");
orderData.put("originIp", "127.0.0.1");
orderData.put("originUserAgent",
    "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_9_4)
    AppleWebKit/537.36 (KHTML, like Gecko) Chrome/46.0.2490.71 Safari/537.36");
orderData.put("trackingCookieName", "tid_blkycggbjquivddbiddxylaufhfdbkj");

orderData.put("trackingCookieValue", "3.Ag.AQ.AQ.AQ.AQ.t..l.AQ.n.VO4AkA.VO4AkA.G89J");
orderData.put("shippingAmount", 0);
orderData.put("shippingDate", "2015-01-02");
orderData.put("shippingDetails", "Free next day shipping");
orderData.put("shippingTrackingUrl", "http://www.shipping.com/1234");

// Add the Order
Response orderResponse = client.target(BRONTO_HOST)
    .path(ADD_ORDER_PATH)
    .queryParam(CREATE_CONTACT, true)
    .queryParam(IGNORE_INVALID_TRACKING, true)
    .request(MediaType.APPLICATION_JSON)
    .header("Authorization", "Bearer " + accessToken.toString())
    .post(Entity.json(orderData));

if (orderResponse.getStatus() !=
Response.Status.CREATED.getStatusCode()) {
    String reason = orderResponse.getHeaderString(REASON_HEADER);
    throw new RuntimeException("Unable to add Order. Reason=" + reason);
}

// Retrieve the Order from the response
Map<String, Object> order = orderResponse.readEntity(Map.class);
System.out.println(order);
}
}

```

Delete Order

Deletes an order with the given Bronto-generated order id.

Overview

If you want to update an order using the order ID generated by your system, use Delete Order By Customer Order Id instead.

This request requires an access token with “orders/carts-write” scope. A 204 No Content response will always be returned.

URI

```
DELETE: https://rest.bronto.com/orders/{orderId}
```

Parameters

Parameter	Type	Description
orderId	Required String	The Bronto-generated unique order ID of the order to be deleted.

Java Code Example

```
import java.util.Map;
import java.util.UUID;

import javax.ws.rs.client.Client;
import javax.ws.rs.client.ClientBuilder;
import javax.ws.rs.client.Entity;
import javax.ws.rs.core.Form;
import javax.ws.rs.core.MediaType;
import javax.ws.rs.core.Response;

public class DeleteOrderExample {

    // Host
    private static final String BRONTO_HOST = "http://rest.bronto.com";
    private static final String BRONTO_AUTH_PATH = "https://auth.bronto.com/
oauth2/token";

    // Paths
    private static final String DELETE_ORDER_PATH = "orders/{orderId}";
    private static final String ORDER_ID = "orderId";

    // OAuth Request property names
    private static final String GRANT_TYPE = "grant_type";
    private static final String CLIENT_ID = "client_id";
    private static final String CLIENT_SECRET = "client_secret";

    // OAuth Request property values
    private static final String CLIENT_CREDENTIALS = "client_credentials";
    private static final String EXAMPLE_CLIENT_ID = "XXXXXXXXXXXXXXXXXXXXXXXXX";
    private static final String EXAMPLE_CLIENT_SECRET
= "XXXXXXXXXXXXXXXXXXXXXXXXX";

    private static final String ACCESS_TOKEN = "access_token";

    private static final String REASON_HEADER = "X-Reason";

    // Id of the Order we are deleting
    private static final String EXAMPLE_ORDER_ID
= "0ad24370-96de-4922-9a53-954530fcbb64";

    public static void main(String[] args) {
        Client client = ClientBuilder.newClient();

        // To be able to access Orders Rest API, you need an access token.
        // First, we build the request data needed to gain an access token
```



```

Form requestData = new Form();
requestData.param(GRANT_TYPE, CLIENT_CREDENTIALS);
requestData.param(CLIENT_ID, EXAMPLE_CLIENT_ID);
requestData.param(CLIENT_SECRET, EXAMPLE_CLIENT_SECRET);

// Then build and send the request
Response oauthResponse = client.target(BRONTO_AUTH_PATH)
    .request(MediaType.APPLICATION_JSON)
    .accept(MediaType.TEXT_PLAIN_TYPE)
    .post(Entity.form(requestData));

if (oauthResponse.getStatus() != Response.Status.OK.getStatusCode()) {
    throw new RuntimeException("Unable to get access token.");
}

// Retrieve the access token from the response
Map<String, Object> responseData = oauthResponse.readEntity(Map.class);
UUID accessToken = UUID.fromString((String)
responseData.get(ACCESS_TOKEN));

// Now to delete the Order
Response orderResponse = client.target(BRONTO_HOST)
    .path(DELETE_ORDER_PATH)
    .resolveTemplate(ORDER_ID, EXAMPLE_ORDER_ID)
    .request(MediaType.APPLICATION_JSON)
    .header("Authorization", "Bearer " + accessToken.toString())
    .delete();

if (orderResponse.getStatus() !=
Response.Status.NO_CONTENT.getStatusCode()) {
    String reason = orderResponse.getHeaderString(REASON_HEADER);
    throw new RuntimeException("Unable to delete Order. Reason=" +
reason);
}
}
}

```

Delete Order By Customer Order Id

Deletes an order with the given customerOrderID.

Overview

The customerOrderID is an ID generated outside of Bronto, typically by your system. If you want to delete an order using the Bronto-generated order ID, use Delete Order instead.

This request requires an access token with “orders/carts-write” scope. A 204 No Content response will always be returned.

URI

```
DELETE: https://rest.bronto.com/orders/customerOrderId/{customerOrderId}
```

Parameters

Parameter	Type	Description
customerOrderId	Required String	The order ID that was generated by your system for the order you want to delete.

Java Code Example

```

package com.bronto.example;

import java.util.Map;
import java.util.UUID;

import javax.ws.rs.client.Client;
import javax.ws.rs.client.ClientBuilder;
import javax.ws.rs.client.Entity;
import javax.ws.rs.core.Form;
import javax.ws.rs.core.MediaType;
import javax.ws.rs.core.Response;

public class DeleteOrderByCustomerIdExample {

    // Host
    private static final String BRONTO_HOST = "http://rest.bronto.com";
    private static final String BRONTO_AUTH_PATH = "https://auth.bronto.com/oauth2/token";

    // Paths
    private static final String DELETE_ORDER_PATH = "orders/customerOrderId/{customerOrderId}";
    private static final String CUSTOMER_ORDER_ID = "customerOrderId";

    // OAuth Request property names
    private static final String GRANT_TYPE = "grant_type";
    private static final String CLIENT_ID = "client_id";
    private static final String CLIENT_SECRET = "client_secret";

    // OAuth Request property values
    private static final String CLIENT_CREDENTIALS = "client_credentials";
    private static final String EXAMPLE_CLIENT_ID = "XXXXXXXXXXXXXXXXXXXXXXXXXXXX";
    private static final String EXAMPLE_CLIENT_SECRET = "XXXXXXXXXXXXXXXXXXXXXXXXXXXX";

    private static final String ACCESS_TOKEN = "access_token";

    private static final String REASON_HEADER = "X-Reason";

    // Id of the Order we are deleting
    private static final String EXAMPLE_ORDER_ID = "MyOrderIdGoesHere";

    public static void main(String[] args) {
        Client client = ClientBuilder.newClient();

        // To be able to access Orders Rest API, you need an access token.
        // First, we build the request data needed to gain an access token
        Form requestData = new Form();
        requestData.param(GRANT_TYPE, CLIENT_CREDENTIALS);
        requestData.param(CLIENT_ID, EXAMPLE_CLIENT_ID);
        requestData.param(CLIENT_SECRET, EXAMPLE_CLIENT_SECRET);

        // Then build and send the request
        Response oauthResponse = client.target(BRONTO_AUTH_PATH)
            .request(MediaType.APPLICATION_JSON)
            .accept(MediaType.TEXT_PLAIN_TYPE)
            .post(Entity.form(requestData));
    }
}

```

```

if (oauthResponse.getStatus() != Response.Status.OK.getStatusCode()) {
    throw new RuntimeException("Unable to get access token.");
}

// Retrieve the access token from the response
Map<String, Object> responseData = oauthResponse.readEntity(Map.class);
UUID accessToken = UUID.fromString((String)
responseData.get(ACCESS_TOKEN));

// Now to delete the Order
Response orderResponse = client.target(BRONTO_HOST)
    .path(DELETE_ORDER_PATH)
    .resolveTemplate(CUSTOMER_ORDER_ID, EXAMPLE_ORDER_ID)
    .request(MediaType.APPLICATION_JSON)
    .header("Authorization", "Bearer " + accessToken.toString())
    .delete();

if (orderResponse.getStatus() !=
Response.Status.NO_CONTENT.getStatusCode()) {
    String reason = orderResponse.getHeaderString(REASON_HEADER);
    throw new RuntimeException("Unable to delete Order. Reason=" +
reason);
}
}
}

```

Get Order

Update an existing order using the Bronto-generated order ID.

Overview

If you want to update an order using the order ID generated by your system, use Update Order By Customer Order Id. Properties that are not included in the request body will remain unchanged on the existing order. No properties are required in the update.

This request accepts both representations of an order's state via the "states" and "status" fields. Updates to either field will be reflected in the other. The "status" field is currently deprecated and it is recommended that you use the "states" field going forward. If both "status" and "states" are both specified, the data supplied in "status" will be used.

You can provide tracking attribution by including a tid with your request or a TrackingCookieName and TrackingCookieValue pair. For more information see Orders REST API Tracking Attribution.



Note: If you have a workflow that is triggered by an Order Is Added trigger node, that workflow will only be triggered for an order the first time an order's status is changed to PROCESSED. This is true both when a PENDING order in Bronto transitions to PROCESSED or if an order with the PROCESSED status is added directly to Bronto.

This request requires an access token with "orders/carts-write" scope. If the request is successful, a 200 Ok response will be returned with the latest version of the Order in the response body. If there is a problem with any of the data in the request body, a 400 Bad Request response will be returned with an explanation of the error. If the `orderDate` is earlier than January 1st, 2000 or more than 5 years in the future, then the order request is rejected with a 400 Bad Request response.

URI

```

POST: https://rest.bronto.com/orders/{orderId}?
createContact={boolean}&ignoreInvalidTid={boolean}

```

Parameters

Parameter	Type	Description
orderId	String	The Bronto-generated unique order ID for the order you want to update.
createContact	Optional Boolean	Determines behavior when the request contains an email address that is not associated with a Contact. If true, a transactional Contact will be created. If false, the service will return a 409 Conflict response. The default value is false.
triggerEvents	Optional Boolean	Determines whether workflows or other contact interactions can be triggered by the state of this order. If false, workflow nodes do not fire. The default value is true.
force	Optional Boolean	Determines whether workflows or other contact interactions will be triggered by the state of this order, regardless of whether the state has changed. If true, an order that is currently processed will re-trigger, potentially re-triggering a workflow that might message the customer. If false, workflows and other contact interactions are only triggered when the state changes, such as a pending order changing to processed.
ignoreInvalidTid	Optional Boolean	Determines behavior when the request contains an invalid tid or TrackingCookieName and TrackingCookieValue pair. If true, the service will ignore the invalid values and continue.

Response Body

A response containing the order with the given ID.

```
{
  emailAddress:validly formatted email address
  contactId:string
  orderDate:ISO-8601 datetime
  status:PENDING | PROCESSED
  hasTracking:boolean
  trackingCookieName:string
  trackingCookieValue:string
  deliveryId:string
  customerOrderId:string
  discountAmount:number
  grandTotal:number
  lineItems:[
    {
      name:string
```

```

        other:string
        sku:string
        category:string
        imageUrl:string
        productUrl:string
        quantity:number
        salePrice:number
        totalPrice:number
        unitPrice:number
        description:string
        position:number
    }
]
originIp:IPv4 or IPv6 address
messageId:string
originUserAgent:string
shippingAmount:number
shippingDate:ISO-8601 datetime
shippingDetails:string
shippingTrackingUrl:string
subtotal:number
taxAmount:number
cartId:UUID
createdDate:ISO-8601 datetime
updatedDate:ISO-8601 datetime
currency:ISO-4217 currency code
states: {
    processed:boolean
    shipped:boolean
}
orderId:UUID
}

```

Java Code Example

```

import java.util.Map;
import java.util.UUID;

import javax.ws.rs.client.Client;
import javax.ws.rs.client.ClientBuilder;
import javax.ws.rs.client.Entity;
import javax.ws.rs.core.Form;
import javax.ws.rs.core.MediaType;
import javax.ws.rs.core.Response;

public class GetOrderExample {

    // Host
    private static final String BRONTO_HOST = "http://rest.bronto.com";
    private static final String BRONTO_AUTH_PATH = "https://auth.bronto.com/
oauth2/token";

    // Paths
    private static final String GET_ORDER_PATH = "orders/{orderId}";
    private static final String ORDER_ID = "orderId";

    // OAuth Request property names
    private static final String GRANT_TYPE = "grant_type";
    private static final String CLIENT_ID = "client_id";
    private static final String CLIENT_SECRET = "client_secret";

```

```

// OAuth Request property values
private static final String CLIENT_CREDENTIALS = "client_credentials";
private static final String EXAMPLE_CLIENT_ID = "XXXXXXXXXXXXXXXXXXXXXXXXXXXX";
private static final String EXAMPLE_CLIENT_SECRET
= "XXXXXXXXXXXXXXXXXXXXXXXXXXXX";

private static final String ACCESS_TOKEN = "access_token";

private static final String REASON_HEADER = "X-Reason";

// Id of the Order we are getting
private static final String EXAMPLE_ORDER_ID
= "0ad24370-96de-4922-9a53-954530fcbb64";

public static void main(String[] args) {
    Client client = ClientBuilder.newClient();

    // To be able to access Orders Rest API, you need an access token.
    // First, we build the request data needed to gain an access token
    Form requestData = new Form();
    requestData.param(GRANT_TYPE, CLIENT_CREDENTIALS);
    requestData.param(CLIENT_ID, EXAMPLE_CLIENT_ID);
    requestData.param(CLIENT_SECRET, EXAMPLE_CLIENT_SECRET);

    // Then build and send the request
    Response oauthResponse = client.target(BRONTO_AUTH_PATH)
        .request(MediaType.APPLICATION_JSON)
        .accept(MediaType.TEXT_PLAIN_TYPE)
        .post(Entity.form(requestData));

    if (oauthResponse.getStatus() != Response.Status.OK.getStatusCode()) {
        throw new RuntimeException("Unable to get access token.");
    }

    // Retrieve the access token from the response
    Map<String, Object> responseData = oauthResponse.readEntity(Map.class);
    UUID accessToken = UUID.fromString((String)
responseData.get(ACCESS_TOKEN));

    // Now to get the Order
    Response orderResponse = client.target(BRONTO_HOST)
        .path(GET_ORDER_PATH)
        .resolveTemplate(ORDER_ID, EXAMPLE_ORDER_ID)
        .request(MediaType.APPLICATION_JSON)
        .header("Authorization", "Bearer " + accessToken.toString())
        .get();

    if (orderResponse.getStatus() ==
Response.Status.NOT_FOUND.getStatusCode()) {
        throw new RuntimeException("No Order found with orderId=" +
EXAMPLE_ORDER_ID);
    } else if (orderResponse.getStatus() !=
Response.Status.OK.getStatusCode()) {
        String reason = orderResponse.getHeaderString(REASON_HEADER);
        throw new RuntimeException("Unable to get Order. Reason=" + reason);
    }

    // Retrieve the Order from the response
    Map<String, Object> order = orderResponse.readEntity(Map.class);
    System.out.println(order);
}

```

```
}
```

Search For Orders

Search for orders using a customerOrderId.

Overview

This request requires an access token with “orders/carts-read” scope. If the request is successful and one or more Orders are found, a 200 Ok response will be returned including a list of the Orders in the response body. If no results are found, a 204 No Content response will be returned. If there is a problem with any of the query parameters, a 400 Bad Request response will be returned with an explanation of the error in the X-Reason header.

URI

```
GET: https://rest.bronto.com/orders?
customerOrderId={customerOrderId}&deliveryId={deliveryId}&messageId={messageId}&contactId={contactId}
```

Parameters

Parameter	Type	Description
customerOrderId	Required String	The customer's ID that is associated with the order.
deliveryId	Optional String	The unique identifier for a single delivery of a message associated with the order. You can use readDeliveries to return deliveryId as part of the deliveryObject .
messageId	Optional String	The unique identifier for a single message associated with the order. You can use readMessages to return messageId as part of the messageObject .
contactId	Optional String	The unique identifier for a contact associated with the order. You can use readContacts or readContactsWithLatestUnsubscribeDate to return contactId as part of the contactObject .

Response Body

A response containing a list of the orders found, if any.

```
[
  {
    emailAddress:validly formatted email address
    contactId:string
    orderDate:ISO-8601 datetime
    status:PENDING | PROCESSED
    hasTracking:boolean
    trackingCookieName:string
    trackingCookieValue:string
    deliveryId:string
    customerOrderId:string
    discountAmount:number
  }
]
```

```

grandTotal:number
lineItems:[
  {
    name:string
    other:string
    sku:string
    category:string
    imageUrl:string
    productUrl:string
    quantity:number
    salePrice:number
    totalPrice:number
    unitPrice:number
    description:string
    position:number
  }
]
messageId:string
originIp:IPv4 or IPv6 address
originUserAgent:string
shippingAmount:number
shippingDate:ISO-8601 datetime
shippingDetails:string
shippingTrackingUrl:string
subtotal:number
taxAmount:number
cartId:UUID
createdDate:ISO-8601 datetime
updatedDate:ISO-8601 datetime
currency:ISO-4217 currency code
states: {
  processed:boolean
  shipped:boolean
}
orderId:UUID
}
]

```

Java Code Example

```

import java.util.Collections;
import java.util.List;
import java.util.Map;
import java.util.UUID;

import javax.ws.rs.client.Client;
import javax.ws.rs.client.ClientBuilder;
import javax.ws.rs.client.Entity;
import javax.ws.rs.core.Form;
import javax.ws.rs.core.MediaType;
import javax.ws.rs.core.Response;

public class SearchForOrdersExample {

    // Host
    private static final String BRONTO_HOST = "http://rest.bronto.com";
    private static final String BRONTO_AUTH_PATH = "https://auth.bronto.com/
oauth2/token";

    // Paths
    private static final String SEARCH_ORDERS_PATH = "orders";

```



```

private static final String CUSTOMER_ORDER_ID = "customerOrderId";

// OAuth Request property names
private static final String GRANT_TYPE = "grant_type";
private static final String CLIENT_ID = "client_id";
private static final String CLIENT_SECRET = "client_secret";

// OAuth Request property values
private static final String CLIENT_CREDENTIALS = "client_credentials";
private static final String EXAMPLE_CLIENT_ID = "XXXXXXXXXXXXXXXXXXXXXXXXXXXX";
private static final String EXAMPLE_CLIENT_SECRET
= "XXXXXXXXXXXXXXXXXXXXXXXXXXXX";

private static final String ACCESS_TOKEN = "access_token";

private static final String REASON_HEADER = "X-Reason";

// Id of the Order we are searching for
private static final String EXAMPLE_CUSTOMER_ORDER_ID = "abc-123";

public static void main(String[] args) {
    Client client = ClientBuilder.newClient();

    // To be able to access Orders Rest API, you need an access token.
    // First, we build the request data needed to gain an access token
    Form requestData = new Form();
    requestData.param(GRANT_TYPE, CLIENT_CREDENTIALS);
    requestData.param(CLIENT_ID, EXAMPLE_CLIENT_ID);
    requestData.param(CLIENT_SECRET, EXAMPLE_CLIENT_SECRET);

    // Then build and send the request
    Response oauthResponse = client.target(BRONTO_AUTH_PATH)
        .request(MediaType.APPLICATION_JSON)
        .accept(MediaType.TEXT_PLAIN_TYPE)
        .post(Entity.form(requestData));

    if (oauthResponse.getStatus() != Response.Status.OK.getStatusCode()) {
        throw new RuntimeException("Unable to get access token.");
    }

    // Retrieve the access token from the response
    Map<String, Object> responseData = oauthResponse.readEntity(Map.class);
    UUID accessToken = UUID.fromString((String)
responseData.get(ACCESS_TOKEN));

    // Now to search for Orders
    Response orderResponse = client.target(BRONTO_HOST)
        .path(SEARCH_ORDERS_PATH)
        .queryParams(CUSTOMER_ORDER_ID, EXAMPLE_CUSTOMER_ORDER_ID)
        .request(MediaType.APPLICATION_JSON)
        .header("Authorization", "Bearer " + accessToken.toString())
        .get();

    // Get Orders from the response
    List<Map<String, Object>> orders = null;
    if (orderResponse.getStatus() == Response.Status.OK.getStatusCode()) {
        orders = orderResponse.readEntity(List.class);
    } else if (orderResponse.getStatus() ==
Response.Status.NO_CONTENT.getStatusCode()) {
        orders = Collections.EMPTY_LIST;
    } else {
        String reason = orderResponse.getHeaderString(REASON_HEADER);

```

```

        throw new RuntimeException("Unable to search for Orders. Reason=" +
            reason);
    }

    System.out.println(orders);
}
}

```

Update Order

Update an existing order using the Bronto-generated order ID.

Overview

If you want to update an order using the order ID generated by your system, use Update Order By Customer Order Id. Properties that are not included in the request body will remain unchanged on the existing order. No properties are required in the update.

This request accepts both representations of an order's state via the "states" and "status" fields. Updates to either field will be reflected in the other. The "status" field is currently deprecated and it is recommended that you use the "states" field going forward. If both "status" and "states" are both specified, the data supplied in "status" will be used.

You can provide tracking attribution by including a tid with your request or a TrackingCookieName and TrackingCookieValue pair. For more information see Orders REST API Tracking Attribution.



Note: If you have a workflow that is triggered by an Order Is Added trigger node, that workflow will only be triggered for an order the first time an order's status is changed to PROCESSED. This is true both when a PENDING order in Bronto transitions to PROCESSED or if an order with the PROCESSED status is added directly to Bronto.

This request requires an access token with "orders/carts-write" scope. If the request is successful, a 200 Ok response will be returned with the latest version of the Order in the response body. If there is a problem with any of the data in the request body, a 400 Bad Request response will be returned with an explanation of the error. If the orderDate is earlier than January 1st, 2000 or more than 5 years in the future, then the order request is rejected with a 400 Bad Request response.

URI

POST: `https://rest.bronto.com/orders/{orderId}?createContact={boolean}&ignoreInvalidTid={boolean}`

Parameters

Parameter	Type	Description
orderId	String	The Bronto-generated unique order ID for the order you want to update.
createContact	Optional Boolean	Determines behavior when the request contains an email address that is not associated with a Contact. If true, a transactional Contact will be created. If false, the service will return a 409 Conflict response. The default value is false.

Parameter	Type	Description
triggerEvents	Optional Boolean	Determines whether workflows or other contact interactions can be triggered by the state of this order. If false, workflow nodes do not fire. The default value is true.
force	Optional Boolean	Determines whether workflows or other contact interactions will be triggered by the state of this order, regardless of whether the state has changed. If true, an order that is currently processed will re-trigger, potentially re-triggering a workflow that might message the customer. If false, workflows and other contact interactions are only triggered when the state changes, such as a pending order changing to processed.
ignoreInvalidTid	Optional Boolean	Determines behavior when the request contains an invalid tid or TrackingCookieName and TrackingCookieValue pair. If true, the service will ignore the invalid values and continue.

Request Body

The order data used to update the order.

```
{
  status:PENDING | PROCESSED
  discountAmount:number
  emailAddress:validly formatted email address
  contactId:string
  grandTotal:number
  lineItems:[
    {
      name:string
      description:string
      sku:string
      other:string
      imageUrl:string
      category:string
      productUrl:string
      quantity:number
      salePrice:number
      totalPrice:number
      unitPrice:number
    }
  ]
  messageId:string
  originIp:IPv4 or IPv6 address
  originUserAgent:string
  shippingAmount:number
  shippingDate:ISO-8601 datetime
  shippingDetails:string
  shippingTrackingUrl:string
}
```

```

    subtotal:number
    taxAmount:number
    trackingCookieName:string
    trackingCookieValue:string
    deliveryId:string
    orderDate:ISO-8601 datetime
    currency:ISO-4217 currency code
    tid:string
    states: {
      processed:boolean
      shipped:boolean
    }
  }
}

```

Response

A response containing the updated order.

```

{
  emailAddress:validly formatted email address
  contactId:string
  orderDate:ISO-8601 datetime
  status:PENDING | PROCESSED
  hasTracking:boolean
  trackingCookieName:string
  trackingCookieValue:string
  deliveryId:string
  customerOrderId:string
  discountAmount:number
  grandTotal:number
  lineItems:[
    {
      name:string
      other:string
      sku:string
      category:string
      imageUrl:string
      productUrl:string
      quantity:number
      salePrice:number
      totalPrice:number
      unitPrice:number
      description:string
      position:number
    }
  ]
  messageId:string
  originIp:IPv4 or IPv6 address
  originUserAgent:string
  shippingAmount:number
  shippingDate:ISO-8601 datetime
  shippingDetails:string
  shippingTrackingUrl:string
  subtotal:number
  taxAmount:number
  cartId:UUID
  createdAt:ISO-8601 datetime
  updatedAt:ISO-8601 datetime
  currency:ISO-4217 currency code
  states: {
    processed:boolean
    shipped:boolean
  }
}

```

```

        orderId:UUID
    }

```

Java Code Example

```

import java.util.Arrays;
import java.util.HashMap;
import java.util.Map;
import java.util.UUID;

import javax.ws.rs.client.Client;
import javax.ws.rs.client.ClientBuilder;
import javax.ws.rs.client.Entity;
import javax.ws.rs.core.Form;
import javax.ws.rs.core.MediaType;
import javax.ws.rs.core.Response;

public class UpdateOrderExample {

    // Host
    private static final String BRONTO_HOST = "http://rest.bronto.com";
    private static final String BRONTO_AUTH_PATH = "https://auth.bronto.com/
oauth2/token";

    // Paths
    private static final String UPDATE_ORDER_PATH = "orders/{orderId}";
    private static final String ORDER_ID = "orderId";
    private static final String CREATE_CONTACT = "createContact";
    private static final String IGNORE_INVALID_TRACKING
= "ignoreInvalidTracking";

    // OAuth Request property names
    private static final String GRANT_TYPE = "grant_type";
    private static final String CLIENT_ID = "client_id";
    private static final String CLIENT_SECRET = "client_secret";

    // OAuth Request property values
    private static final String CLIENT_CREDENTIALS = "client_credentials";
    private static final String EXAMPLE_CLIENT_ID = "xxxxxxxxxxxxxxxxxxxxxxxx";
    private static final String EXAMPLE_CLIENT_SECRET
= "xxxxxxxxxxxxxxxxxxxxxxxx";

    private static final String ACCESS_TOKEN = "access_token";

    private static final String REASON_HEADER = "X-Reason";

    // Id of the Order we are updating
    private static final String EXAMPLE_ORDER_ID
= "0ad24370-96de-4922-9a53-954530fcbb64";

    public static void main(String[] args) {
        Client client = ClientBuilder.newClient();

        // To be able to access Orders Rest API, you need an access token.
        // First, we build the request data needed to gain an access token
        Form requestData = new Form();
        requestData.param(GRANT_TYPE, CLIENT_CREDENTIALS);
        requestData.param(CLIENT_ID, EXAMPLE_CLIENT_ID);
        requestData.param(CLIENT_SECRET, EXAMPLE_CLIENT_SECRET);
    }

```

```

// Then build and send the request
Response oauthResponse = client.target(BRONTO_AUTH_PATH)
    .request(MediaType.APPLICATION_JSON)
    .accept(MediaType.TEXT_PLAIN_TYPE)
    .post(Entity.form(requestData));

if (oauthResponse.getStatus() != Response.Status.OK.getStatusCode()) {
    throw new RuntimeException("Unable to get access token.");
}

// Retrieve the access token from the response
Map<String, Object> responseData =
oauthResponse.readEntity(Map.class);
UUID accessToken = UUID.fromString((String)
responseData.get(ACCESS_TOKEN));

// Now to update the Order, we first build the request data
Map<String, Object> lineItemData = new HashMap<String,
Object>(2);
lineItemData.put("name", "Coffee Mug");
lineItemData.put("description", "A cool coffee mug");
lineItemData.put("category", "Kitchen Stuffs");
lineItemData.put("other", "5oz");
lineItemData.put("imageUrl", "http://www.example.com/
images/0003105.jpg");
lineItemData.put("productUrl", "http://www.example.com/
products/0003105");
lineItemData.put("sku", "0003105");
lineItemData.put("quantity", 1);
lineItemData.put("totalPrice", 3.99f);
lineItemData.put("unitPrice", 3.99f);
lineItemData.put("salePrice", 3.99f);

Map<String, Object> orderData = new HashMap<String, Object>(
2);
orderData.put("emailAddress", "example@email.com");
orderData.put("currency", "USD");
orderData.put("grandTotal", 4.27f);
orderData.put("discountAmount", 0);
orderData.put("taxAmount", 0.28f);
orderData.put("subtotal", 3.99f);
orderData.put("lineItems", Arrays.asList(lineItemData));
orderData.put("status", "PROCESSED");
orderData.put("originIp", "127.0.0.1");
orderData.put("originUserAgent",
    "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_9_4)
AppleWebKit/537.36 (KHTML, like Gecko) Chrome/46.0.2490.71 Safari/537.36");
orderData.put("trackingCookieName", "tid_blkycggbjquivddbiddxylaufhfdbkj");

orderData.put("trackingCookieValue", "3.Ag.AQ.AQ.AQ.AQ.t..1.AQ.n.VO4Aka.VO4Aka.G89J");
orderData.put("shippingAmount", 0);
orderData.put("shippingDate", "2015-01-02");
orderData.put("shippingDetails", "Free next day shipping");
orderData.put("shippingTrackingUrl", "http://www.shipping.com/1234");

// Update the Order
Response orderResponse = client.target(BRONTO_HOST)
    .path(UPDATE_ORDER_PATH)
    .resolveTemplate(ORDER_ID, EXAMPLE_ORDER_ID)
    .queryParam(CREATE_CONTACT, true)

```

```

        .queryParams(IGNORE_INVALID_TRACKING, true)
        .request(MediaType.APPLICATION_JSON)
        .header("Authorization", "Bearer " + accessToken.toString())
        .post(Entity.json(orderData));

        if (orderResponse.getStatus() ==
Response.Status.NOT_FOUND.getStatusCode()) {
            throw new RuntimeException("No Order found with orderId=" +
EXAMPLE_ORDER_ID);
        } else if (orderResponse.getStatus() !=
Response.Status.OK.getStatusCode()) {
            String reason = orderResponse.getHeaderString(REASON_HEADER);
            throw new RuntimeException("Unable to update Order. Reason=" +
reason);
        }

        // Retrieve the Order from the response
        Map<String, Object> order = orderResponse.readEntity(Map.class);
        System.out.println(order);
    }
}

```

Update Order By Customer Order Id

Updates an existing order using the customerOrderId.

Overview

The customerOrderId is an ID generated outside of Bronto, typically by your system. If you want to update an order using the Bronto-generated order ID, use Update Order instead. Properties that are not included in the request body will remain unchanged on the existing order. No properties are required in the update.

This request accepts both representations of an order's state via the "states" and "status" fields. Updates to either field will be reflected in the other. The "status" field is currently deprecated and it is recommended that you use the "states" field going forward. If both "status" and "states" are both specified, the data supplied in "status" will be used.

You can provide tracking attribution by including a tid with your request or a TrackingCookieName and TrackingCookieValue pair. For more information see Orders REST API Tracking Attribution.



Note: If you have a workflow that is triggered by an Order Is Added trigger node, that workflow will only be triggered for an order the first time an order's status is changed to PROCESSED. This is true both when a PENDING order in Bronto transitions to PROCESSED or if an order with the PROCESSED status is added directly to Bronto.

This request requires an access token with "orders/carts-write" scope. If the request is successful, a 200 Ok response will be returned with the latest version of the Order in the response body. If there is a problem with any of the data in the request body, a 400 Bad Request response will be returned with an explanation of the error. If the orderDate is earlier than January 1st, 2000 or more than 5 years in the future, then the order request is rejected with a 400 Bad Request response.

URI

```

POST: https://rest.bronto.com/orders/customerOrderId/{customerOrderId}?
createContact={boolean}&ignoreInvalidTid={boolean}

```

Parameters

Parameter	Type	Description
customerOrderId	Required String	The ID generated by your system for the order you want to update.

Parameter	Type	Description
createContact	Optional Boolean	Determines behavior when the request contains an email address that is not associated with a Contact. If true, a transactional Contact will be created. If false, the service will return a 409 Conflict response. The default value is false.
triggerEvents	Optional Boolean	Determines whether workflows or other contact interactions can be triggered by the state of this order. If false, workflow nodes do not fire. The default value is true.
force	Optional Boolean	Determines whether workflows or other contact interactions will be triggered by the state of this order, regardless of whether the state has changed. If true, an order that is currently processed will re-trigger, potentially re-triggering a workflow that might message the customer. If false, workflows and other contact interactions are only triggered when the state changes, such as a pending order changing to processed.
ignoreInvalidTid	Optional Boolean	Determines behavior when the request contains an invalid TID or TrackingCookieName and TrackingCookieValue pair. If true, the service will ignore the invalid values and continue.

Request Body

The order data used to update the order.

```
{
  status:PENDING | PROCESSED
  discountAmount:number
  emailAddress:validly formatted email address
  contactId:string
  deliveryId:string
  grandTotal:number
  lineItems:[
    {
      name:string
      description:string
      sku:string
      other:string
      imageUrl:string
      category:string
      productUrl:string
      quantity:number
      salePrice:number
      totalPrice:number
      unitPrice:number
    }
  ]
}
```



```

    }
  ]
  originIp:IPv4 or IPv6 address
  messageId:string
  originUserAgent:string
  shippingAmount:number
  shippingDate:ISO-8601 datetime
  shippingDetails:string
  shippingTrackingUrl:string
  subtotal:number
  taxAmount:number
  trackingCookieName:string
  trackingCookieValue:string
  tid:string
  orderDate:ISO-8601 datetime
  currency:ISO-4217 currency code
  states: {
    processed:boolean
    shipped:boolean
  }
}

```

Response

A response containing the updated order.

```

{
  emailAddress:validly formatted email address
  contactId:string
  orderDate:ISO-8601 datetime
  status:PENDING | PROCESSED
  hasTracking:boolean
  trackingCookieName:string
  trackingCookieValue:string
  deliveryId:string
  customerOrderId:string
  discountAmount:number
  grandTotal:number
  lineItems:[
    {
      name:string
      other:string
      sku:string
      category:string
      imageUrl:string
      productUrl:string
      quantity:number
      salePrice:number
      totalPrice:number
      unitPrice:number
      description:string
      position:number
    }
  ]
  originIp:IPv4 or IPv6 address
  messageId:string
  originUserAgent:string
  shippingAmount:number
  shippingDate:ISO-8601 datetime
  shippingDetails:string
  shippingTrackingUrl:string
  subtotal:number
  taxAmount:number
}

```

```

    cartId:UUID
    createdAt:ISO-8601 datetime
    updatedAt:ISO-8601 datetime
    currency:ISO-4217 currency code
    states: {
        processed:boolean
        shipped:boolean
    }
    orderId:UUID
}

```

Java Code Example

```

import java.util.Arrays;
import java.util.HashMap;
import java.util.Map;
import java.util.UUID;

import javax.ws.rs.client.Client;
import javax.ws.rs.client.ClientBuilder;
import javax.ws.rs.client.Entity;
import javax.ws.rs.core.Form;
import javax.ws.rs.core.MediaType;
import javax.ws.rs.core.Response;

public class UpdateOrderByCustomerIdExample {

    // Host
    private static final String BRONTO_HOST = "http://rest.bronto.com";
    private static final String BRONTO_AUTH_PATH = "https://auth.bronto.com/
oauth2/token";

    // Paths
    private static final String UPDATE_ORDER_PATH = "orders/customerOrderId/
{customerOrderId}";
    private static final String CUSTOMER_ORDER_ID = "customerOrderId";
    private static final String CREATE_CONTACT = "createContact";
    private static final String IGNORE_INVALID_TRACKING
= "ignoreInvalidTracking";

    // OAuth Request property names
    private static final String GRANT_TYPE = "grant_type";
    private static final String CLIENT_ID = "client_id";
    private static final String CLIENT_SECRET = "client_secret";

    // OAuth Request property values
    private static final String CLIENT_CREDENTIALS = "client_credentials";
    private static final String EXAMPLE_CLIENT_ID = "xxxxxxxxxxxxxxxxxxxxxxxx";
    private static final String EXAMPLE_CLIENT_SECRET
= "xxxxxxxxxxxxxxxxxxxxxxxx";

    private static final String ACCESS_TOKEN = "access_token";

    private static final String REASON_HEADER = "X-Reason";

    // Id of the Order we are updating
    private static final String EXAMPLE_ORDER_ID = "MyOrderIdGoesHere";

    public static void main(String[] args) {

```

```

Client client = ClientBuilder.newClient();

// To be able to access Orders Rest API, you need an access token.
// First, we build the request data needed to gain an access token
Form requestData = new Form();
requestData.param(GRANT_TYPE, CLIENT_CREDENTIALS);
requestData.param(CLIENT_ID, EXAMPLE_CLIENT_ID);
requestData.param(CLIENT_SECRET, EXAMPLE_CLIENT_SECRET);

// Then build and send the request
Response oauthResponse = client.target(BRANTO_AUTH_PATH)
    .request(MediaType.APPLICATION_JSON)
    .accept(MediaType.TEXT_PLAIN_TYPE)
    .post(Entity.form(requestData));

if (oauthResponse.getStatus() != Response.Status.OK.getStatusCode()) {
    throw new RuntimeException("Unable to get access token.");
}

// Retrieve the access token from the response
Map<String, Object> responseData =
oauthResponse.readEntity(Map.class);
UUID accessToken = UUID.fromString((String)
responseData.get(ACCESS_TOKEN));

// Now to update the Order, we first build the request data
Map<String, Object> lineItemData = new HashMap<String,
Object>(2);
lineItemData.put("name", "Coffee Mug");
lineItemData.put("description", "A cool coffee mug");
lineItemData.put("category", "Kitchen Stuffs");
lineItemData.put("other", "5oz");
lineItemData.put("imageUrl", "http://www.example.com/
images/0003105.jpg");
lineItemData.put("productUrl", "http://www.example.com/
products/0003105");
lineItemData.put("sku", "0003105");
lineItemData.put("quantity", 1);
lineItemData.put("totalPrice", 3.99f);
lineItemData.put("unitPrice", 3.99f);
lineItemData.put("salePrice", 3.99f);

Map<String, Object> orderData = new HashMap<String, Object>(
2);
orderData.put("emailAddress", "example@email.com");
orderData.put("currency", "USD");
orderData.put("grandTotal", 4.27f);
orderData.put("discountAmount", 0);
orderData.put("taxAmount", 0.28f);
orderData.put("subtotal", 3.99f);
orderData.put("lineItems", Arrays.asList(lineItemData));
orderData.put("status", "PROCESSED");
orderData.put("originIp", "127.0.0.1");
orderData.put("originUserAgent",
    "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_9_4)
AppleWebKit/537.36 (KHTML, like Gecko) Chrome/46.0.2490.71 Safari/537.36");
orderData.put("trackingCookieName", "tid_blkycggbjquivddbiddxylaufhfdbkj");

orderData.put("trackingCookieValue", "3.Ag.AQ.AQ.AQ.AQ.t..1.AQ.n.VO4AkA.VO4AkA.G89J");
orderData.put("shippingAmount", 0);
orderData.put("shippingDate", "2015-01-02");

```

```

orderData.put("shippingDetails", "Free next day shipping");
orderData.put("shippingTrackingUrl", "http://www.shipping.com/1234");

// Update the Order
Response orderResponse = client.target(BRANTO_HOST)
    .path(UPDATE_ORDER_PATH)
    .resolveTemplate(CUSTOMER_ORDER_ID, EXAMPLE_ORDER_ID)
    .queryParams(CREATE_CONTACT, true)
    .queryParams(IGNORE_INVALID_TRACKING, true)
    .request(MediaType.APPLICATION_JSON)
    .header("Authorization", "Bearer " + accessToken.toString())
    .post(Entity.json(orderData));

if (orderResponse.getStatus() ==
Response.Status.NOT_FOUND.getStatusCode()) {
    throw new RuntimeException("No Order found with customerId=" +
EXAMPLE_ORDER_ID);
} else if (orderResponse.getStatus() !=
Response.Status.OK.getStatusCode()) {
    String reason = orderResponse.getHeaderString(REASON_HEADER);
    throw new RuntimeException("Unable to update Order. Reason=" +
reason);
}

// Retrieve the Order from the response
Map<String, Object> order = orderResponse.readEntity(Map.class);
System.out.println(order);
}
}

```

Abandon Cart

Abandon a cart with the given ID. The cart must not be in the COMPLETE or EXPIRED status. This sets the cart's status to ABANDONED.

Overview

This request requires an access token with “orders/carts-write” scope. If the request is successful, a 204 No Content response will be returned. If no Cart can be found with the given cartId, a 404 Not Found response will be returned. If the Cart's status is EXPIRED or COMPLETE, a 409 Conflict response will be returned.

URI

PUT: <https://rest.bronto.com/carts/{cartId}/abandon>

Parameters

Parameter	Type	Description
cartId	Required String	The unique ID of the cart.

Java Code Example

```

import java.util.Map;
import java.util.UUID;

import javax.ws.rs.client.Client;
import javax.ws.rs.client.ClientBuilder;
import javax.ws.rs.client.Entity;
import javax.ws.rs.core.Form;

```

```

import javax.ws.rs.core.MediaType;
import javax.ws.rs.core.Response;

public class AbandonCartExample {

    // Host
    private static final String BRONTO_HOST = "http://rest.bronto.com";
    private static final String BRONTO_AUTH_PATH = "https://auth.bronto.com/
oauth2/token";

    // Paths
    private static final String ABANDON_CART_PATH = "carts/{cartId}/abandon";
    private static final String CART_ID = "cartId";

    // OAuth Request property names
    private static final String GRANT_TYPE = "grant_type";
    private static final String CLIENT_ID = "client_id";
    private static final String CLIENT_SECRET = "client_secret";

    // OAuth Request property values
    private static final String CLIENT_CREDENTIALS = "client_credentials";
    private static final String EXAMPLE_CLIENT_ID = "XXXXXXXXXXXXXXXXXXXXXXXXXXXX";
    private static final String EXAMPLE_CLIENT_SECRET
= "XXXXXXXXXXXXXXXXXXXXXXXXXXXX";

    private static final String ACCESS_TOKEN = "access_token";

    private static final String REASON_HEADER = "X-Reason";

    // Id of the Cart we are abandoning
    private static final String EXAMPLE_CART_ID = "7255c31c-39ac-4611-
a2b7-974b0ebfa555";

    public static void main(String[] args) {
        Client client = ClientBuilder.newClient();

        // To be able to access Orders Rest API, you need an access token.
        // First, we build the request data needed to gain an access token
        Form requestData = new Form();
        requestData.param(GRANT_TYPE, CLIENT_CREDENTIALS);
        requestData.param(CLIENT_ID, EXAMPLE_CLIENT_ID);
        requestData.param(CLIENT_SECRET, EXAMPLE_CLIENT_SECRET);

        // Then build and send the request
        Response oauthResponse = client.target(BRONTO_AUTH_PATH)
            .request(MediaType.APPLICATION_JSON)
            .accept(MediaType.TEXT_PLAIN_TYPE)
            .post(Entity.form(requestData));

        if (oauthResponse.getStatus() != Response.Status.OK.getStatusCode()) {
            throw new RuntimeException("Unable to get access token.");
        }

        // Retrieve the access token from the response
        Map<String, Object> responseData = oauthResponse.readEntity(Map.class);
        UUID accessToken = UUID.fromString((String)
responseData.get(ACCESS_TOKEN));

        // Now to abandon the Cart
        Response cartResponse = client.target(BRONTO_HOST)
            .path(ABANDON_CART_PATH)

```

```

        .resolveTemplate(CART_ID, EXAMPLE_CART_ID)
        .request(MediaType.APPLICATION_JSON)
        .header("Authorization", "Bearer " + accessToken.toString())
        .put(Entity.json("")); // No body needed

    if (cartResponse.getStatus() !=
        Response.Status.NO_CONTENT.getStatusCode()) {
        String reason = cartResponse.getHeaderString(REASON_HEADER);
        throw new RuntimeException("Unable to abandon Cart. Reason=" +
            reason);
    }
}

```

Add Cart

This call can be used to create a new cart. Only the `customerCartId` field and the `quantity` and `totalPrice` fields of all, if any, `lineItems` are strictly required.

Overview

You can provide tracking attribution by including a `tid` with your request or a `TrackingCookieName` and `TrackingCookieValue` pair. For more information see [Orders REST API Tracking Attribution](#).

This request requires an access token with “orders/carts-write” scope. If the request is successful, a 201 Created response will be returned with the newly created Cart in the response body. If a Cart with the same `customerCartId` already exists, a 409 Conflict response will be returned.

URI

```

POST: https://rest.bronto.com/carts?
createContact={boolean}&ignoreInvalidTid={boolean}

```

Parameters

Parameter	Type	Description
<code>createContact</code>	Optional Boolean	Determines behavior when the request contains an email address that is not associated with a Contact. If true, a transactional Contact will be created. If false, the service will return a 409 Conflict response. The default value is false.
<code>ignoreInvalidTid</code>	Optional Boolean	Determines behavior when the request contains an invalid TID or <code>TrackingCookieName</code> and <code>TrackingCookieValue</code> pair. If true, the service will ignore the invalid values and continue.

Request Body

Cart data used to create the cart.

```

{
  url: string
  customerCartId: string
  discountAmount: number
}

```

```

emailAddress: validly formatted email address
contactId:string
grandTotal: number
deliveryId:string
lineItems: [
  {
    name: string
    description: string
    sku: string
    other: string
    imageUrl: string
    category: string
    productUrl: string
    quantity: number
    salePrice: number
    totalPrice: number
    unitPrice: number
  }
]
originIp: string
messageId:string
originUserAgent: string
shippingAmount: number
shippingDate: ISO-8601 datetime
shippingDetails: string
shippingTrackingUrl: string
subtotal: number
taxAmount: number
trackingCookieName: string
trackingCookieValue: string
tid:string
phase: SHOPPING | BILLING | PAYMENT | SHIPPING_INFO | SHIPPING_METHOD |
ORDER_REVIEW | ORDER_COMPLETE
currency: ISO-4217 currency code
}

```

Response

A response containing the newly created cart.

```

{
  url:string
  emailAddress:validly formatted email address
  contactId:string
  status:ACTIVE | ABANDONED | EXPIRED | COMPLETE
  phase:SHOPPING | BILLING | PAYMENT | SHIPPING_INFO | SHIPPING_METHOD |
ORDER_REVIEW | ORDER_COMPLETE
  hasTracking:boolean
  customerCartId:string
  trackingCookieName:string
  trackingCookieValue:string
  deliveryId:string
  discountAmount:number
  grandTotal:number
  lineItems:[
    {
      name:string
      other:string
      sku:string
      category:string
      imageUrl:string
      productUrl:string
      quantity:number
    }
  ]
}

```

```

        salePrice:number
        totalPrice:number
        unitPrice:number
        description:string
        position:number
    }
]
originIp:IPv4 or IPv6 address
messageId:string
originUserAgent:string
shippingAmount:number
shippingDate:ISO-8601 datetime
shippingDetails:string
shippingTrackingUrl:string
subtotal:number
taxAmount:number
cartId:UUID
createdDate:ISO-8601 datetime
updatedDate:ISO-8601 datetime
currency:ISO-4217 currency code
}

```

Java Code Example

```

import java.util.Arrays;
import java.util.HashMap;
import java.util.Map;
import java.util.UUID;

import javax.ws.rs.client.Client;
import javax.ws.rs.client.ClientBuilder;
import javax.ws.rs.client.Entity;
import javax.ws.rs.core.Form;
import javax.ws.rs.core.MediaType;
import javax.ws.rs.core.Response;

public class AddCartExample {

    // Host
    private static final String BRONTO_HOST = "http://rest.bronto.com";
    private static final String BRONTO_AUTH_PATH = "https://auth.bronto.com/
oauth2/token";

    // Paths
    private static final String ADD_CART_PATH = "carts";
    private static final String CREATE_CONTACT = "createContact";
    private static final String IGNORE_INVALID_TRACKING
= "ignoreInvalidTracking";

    // OAuth Request property names
    private static final String GRANT_TYPE = "grant_type";
    private static final String CLIENT_ID = "client_id";
    private static final String CLIENT_SECRET = "client_secret";

    // OAuth Request property values
    private static final String CLIENT_CREDENTIALS = "client_credentials";
    private static final String EXAMPLE_CLIENT_ID = "xxxxxxxxxxxxxxxxxxxxxxxx";
    private static final String EXAMPLE_CLIENT_SECRET
= "xxxxxxxxxxxxxxxxxxxxxxxx";

```



```

private static final String ACCESS_TOKEN = "access_token";

private static final String REASON_HEADER = "X-Reason";

public static void main(String[] args) {
    Client client = ClientBuilder.newClient();

    // To be able to access Orders Rest API, you need an access token.
    // First, we build the request data needed to gain an access token
    Form requestData = new Form();
    requestData.param(GRANT_TYPE, CLIENT_CREDENTIALS);
    requestData.param(CLIENT_ID, EXAMPLE_CLIENT_ID);
    requestData.param(CLIENT_SECRET, EXAMPLE_CLIENT_SECRET);

    // Then build and send the request
    Response oauthResponse = client.target(BRONTO_AUTH_PATH)
        .request(MediaType.APPLICATION_JSON)
        .accept(MediaType.TEXT_PLAIN_TYPE)
        .post(Entity.form(requestData));

    if (oauthResponse.getStatus() != Response.Status.OK.getStatusCode()) {
        throw new RuntimeException("Unable to get access token.");
    }

    // Retrieve the access token from the response
    Map<String, Object> responseData = oauthResponse.readEntity(Map.class);
    UUID accessToken = UUID.fromString((String)
responseData.get(ACCESS_TOKEN));

    // Now to add a Cart, we first build the request data
    Map<String, Object> lineItemData = new HashMap<String, Object>(2);
    lineItemData.put("name", "Coffee Mug");
    lineItemData.put("description", "A cool coffee mug");
    lineItemData.put("category", "Kitchen Stuffs");
    lineItemData.put("other", "5oz");
    lineItemData.put("imageUrl", "http://www.example.com/
images/0003105.jpg");
    lineItemData.put("productUrl", "http://www.example.com/
products/0003105");
    lineItemData.put("sku", "0003105");
    lineItemData.put("quantity", 1);
    lineItemData.put("totalPrice", 3.99f);
    lineItemData.put("unitPrice", 3.99f);
    lineItemData.put("salePrice", 3.99f);

    Map<String, Object> cartData = new HashMap<String, Object>(2);
    cartData.put("customerCartId", "abc-123");
    cartData.put("emailAddress", "example@email.com");
    cartData.put("currency", "USD");
    cartData.put("grandTotal", 4.27f);
    cartData.put("discountAmount", 0);
    cartData.put("taxAmount", 0.28f);
    cartData.put("subtotal", 3.99f);
    cartData.put("lineItems", Arrays.asList(lineItemData));
    cartData.put("phase", "SHOPPING");
    cartData.put("url", "http://www.example.com/carts/abc-123");
    cartData.put("originIp", "127.0.0.1");
    cartData.put("originUserAgent",
        "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_9_4)
AppleWebKit/537.36 (KHTML, like Gecko) Chrome/46.0.2490.71 Safari/537.36");

```

```

cartData.put("trackingCookieName", "tid_bkycggbjquivddbiddxylaufhfdbkj");

cartData.put("trackingCookieValue", "3.Ag.AQ.AQ.AQ.AQ.t..l.AQ.n.VO4AkA.VO4AkA.G89Jov");
cartData.put("shippingAmount", 0);
cartData.put("shippingDate", "2015-01-02");
cartData.put("shippingDetails", "Free next day shipping");
cartData.put("shippingTrackingUrl", "http://www.shipping.com/1234");

// Add the Cart
Response cartResponse = client.target(BRANTO_HOST)
    .path(ADD_CART_PATH)
    .queryParam(CREATE_CONTACT, true)
    .queryParam(IGNORE_INVALID_TRACKING, true)
    .request(MediaType.APPLICATION_JSON)
    .header("Authorization", "Bearer " + accessToken.toString())
    .post(Entity.json(cartData));

if (cartResponse.getStatus() != Response.Status.CREATED.getStatusCode())
{
    String reason = cartResponse.getHeaderString(REASON_HEADER);
    throw new RuntimeException("Unable to add Cart. Reason=" + reason);
}

// Retrieve the Cart from the response
Map<String, Object> cart = cartResponse.readEntity(Map.class);
System.out.println(cart);
}
}

```

Complete Cart

Complete a cart with the given id. This sets the cart's status to COMPLETED and will make the cart immutable.

Overview

This request requires an access token with “orders/carts-write” scope. If the request is successful, a 204 No Content response will be returned. If no Cart can be found with the given cartId, a 404 Not Found response will be returned. If the Cart's status is EXPIRED or COMPLETE, a 409 Conflict response will be returned.

URI

POST: <https://rest.brnto.com/carts/{cartId}/complete>

Parameters

Parameter	Type	Description
cartId	Required String	The unique ID of the cart.

Java Code Example

```

import java.util.Map;
import java.util.UUID;

import javax.ws.rs.client.Client;
import javax.ws.rs.client.ClientBuilder;
import javax.ws.rs.client.Entity;
import javax.ws.rs.core.Form;
import javax.ws.rs.core.MediaType;

```

```

import javax.ws.rs.core.Response;

public class CompleteCartExample {

    // Host
    private static final String BRONTO_HOST = "http://rest.bronto.com";
    private static final String BRONTO_AUTH_PATH = "https://auth.bronto.com/
oauth2/token";

    // Paths
    private static final String COMPLETE_CART_PATH = "carts/{cartId}/
complete";
    private static final String CART_ID = "cartId";

    // OAuth Request property names
    private static final String GRANT_TYPE = "grant_type";
    private static final String CLIENT_ID = "client_id";
    private static final String CLIENT_SECRET = "client_secret";

    // OAuth Request property values
    private static final String CLIENT_CREDENTIALS = "client_credentials";
    private static final String EXAMPLE_CLIENT_ID = "XXXXXXXXXXXXXXXXXXXXXXXXX";
    private static final String EXAMPLE_CLIENT_SECRET =
"XXXXXXXXXXXXXXXXXXXXXXXXX";

    private static final String ACCESS_TOKEN = "access_token";

    private static final String REASON_HEADER = "X-Reason";

    // Id of the Cart we are completing
    private static final String EXAMPLE_CART_ID = "7255c31c-39ac-4611-
a2b7-974b0ebfa555";

    public static void main(String[] args) {
        Client client = ClientBuilder.newClient();

        // To be able to access Orders Rest API, you need an access token.
        // First, we build the request data needed to gain an access token
        Form requestData = new Form();
        requestData.param(GRANT_TYPE, CLIENT_CREDENTIALS);
        requestData.param(CLIENT_ID, EXAMPLE_CLIENT_ID);
        requestData.param(CLIENT_SECRET, EXAMPLE_CLIENT_SECRET);

        // Then build and send the request
        Response oauthResponse = client.target(BRONTO_AUTH_PATH)
            .request(MediaType.APPLICATION_JSON)
            .accept(MediaType.TEXT_PLAIN_TYPE)
            .post(Entity.form(requestData));

        if (oauthResponse.getStatus() != Response.Status.OK.getStatusCode()) {
            throw new RuntimeException("Unable to get access token.");
        }

        // Retrieve the access token from the response
        Map<String, Object> responseData = oauthResponse.readEntity(Map.class);
        UUID accessToken = UUID.fromString((String)
responseData.get(ACCESS_TOKEN));

        // Now to complete the Cart
        Response cartResponse = client.target(BRONTO_HOST)
            .path(COMPLETE_CART_PATH)
            .resolveTemplate(CART_ID, EXAMPLE_CART_ID)

```

```

        .request(MediaType.APPLICATION_JSON)
        .header("Authorization", "Bearer " + accessToken.toString())
        .post(Entity.json("")); // No body needed

    if (cartResponse.getStatus() !=
        Response.Status.NO_CONTENT.getStatusCode()) {
        String reason = cartResponse.getHeaderString(REASON_HEADER);
        throw new RuntimeException("Unable to complete Cart. Reason=" +
            reason);
    }
}
}

```

Delete Cart

Delete a cart with the given id.

Overview

This request requires an access token with “orders/carts-write” scope. A 204 No Content response will always be returned.

URI

```
DELETE: https://rest.bronto.com/carts/{cartId}
```

Parameters

Parameter	Type	Description
cartId	Required String	The unique ID of the cart.

Java Code Example

```

import java.util.Map;
import java.util.UUID;

import javax.ws.rs.client.Client;
import javax.ws.rs.client.ClientBuilder;
import javax.ws.rs.client.Entity;
import javax.ws.rs.core.Form;
import javax.ws.rs.core.MediaType;
import javax.ws.rs.core.Response;

public class DeleteCartExample {

    // Host
    private static final String BRONTO_HOST = "http://rest.bronto.com";
    private static final String BRONTO_AUTH_PATH = "https://auth.bronto.com/
oauth2/token";

    // Paths
    private static final String DELETE_CART_PATH = "carts/{cartId}";
    private static final String CART_ID = "cartId";

    // OAuth Request property names
    private static final String GRANT_TYPE = "grant_type";
    private static final String CLIENT_ID = "client_id";
    private static final String CLIENT_SECRET = "client_secret";

```

```

// OAuth Request property values
private static final String CLIENT_CREDENTIALS = "client_credentials";
private static final String EXAMPLE_CLIENT_ID = "XXXXXXXXXXXXXXXXXXXXXXXXXXXX";
private static final String EXAMPLE_CLIENT_SECRET
= "XXXXXXXXXXXXXXXXXXXXXXXXXXXX";

private static final String ACCESS_TOKEN = "access_token";

private static final String REASON_HEADER = "X-Reason";

// Id of the Cart we are deleting
private static final String EXAMPLE_CART_ID = "7255c31c-39ac-4611-
a2b7-974b0ebfa555";

public static void main(String[] args) {
    Client client = ClientBuilder.newClient();

    // To be able to access Orders Rest API, you need an access token.
    // First, we build the request data needed to gain an access token
    Form requestData = new Form();
    requestData.param(GRANT_TYPE, CLIENT_CREDENTIALS);
    requestData.param(CLIENT_ID, EXAMPLE_CLIENT_ID);
    requestData.param(CLIENT_SECRET, EXAMPLE_CLIENT_SECRET);

    // Then build and send the request
    Response oauthResponse = client.target(BRONTO_AUTH_PATH)
        .request(MediaType.APPLICATION_JSON)
        .accept(MediaType.TEXT_PLAIN_TYPE)
        .post(Entity.form(requestData));

    if (oauthResponse.getStatus() != Response.Status.OK.getStatusCode()) {
        throw new RuntimeException("Unable to get access token.");
    }

    // Retrieve the access token from the response
    Map<String, Object> responseData = oauthResponse.readEntity(Map.class);
    UUID accessToken = UUID.fromString((String)
responseData.get(ACCESS_TOKEN));

    // Now to delete the Cart
    Response cartResponse = client.target(BRONTO_HOST)
        .path(DELETE_CART_PATH)
        .resolveTemplate(CART_ID, EXAMPLE_CART_ID)
        .request(MediaType.APPLICATION_JSON)
        .header("Authorization", "Bearer " + accessToken.toString())
        .delete();

    if (cartResponse.getStatus() !=
Response.Status.NO_CONTENT.getStatusCode()) {
        String reason = cartResponse.getHeaderString(REASON_HEADER);
        throw new RuntimeException("Unable to delete Cart. Reason=" + reason);
    }
}
}

```

Expire Cart

Expire a currently abandoned cart with the given ID. This sets the cart's status to EXPIRED and will make the cart immutable. A cart can only be expired if its status is ABANDONED.

Overview

This request requires an access token with “orders/carts-write” scope. If the request is successful, a 204 No Content response will be returned. If no Cart can be found with the given cartId, a 404 Not Found response will be returned. If the Cart’s status is EXPIRED or COMPLETE, a 409 Conflict response will be returned.

URI

POST: `https://rest.bronto.com/carts/{cartId}/expire`

Parameters

Parameter	Type	Description
cartId	Required String	The unique ID of the cart.

Java Code Example

```
import java.util.Map;
import java.util.UUID;

import javax.ws.rs.client.Client;
import javax.ws.rs.client.ClientBuilder;
import javax.ws.rs.client.Entity;
import javax.ws.rs.core.Form;
import javax.ws.rs.core.MediaType;
import javax.ws.rs.core.Response;

public class ExpireCartExample {

    // Host
    private static final String BRONTO_HOST = "http://rest.bronto.com";
    private static final String BRONTO_AUTH_PATH = "https://auth.bronto.com/oauth2/token";

    // Paths
    private static final String EXPIRE_CART_PATH = "carts/{cartId}/expire";
    private static final String CART_ID = "cartId";

    // OAuth Request property names
    private static final String GRANT_TYPE = "grant_type";
    private static final String CLIENT_ID = "client_id";
    private static final String CLIENT_SECRET = "client_secret";

    // OAuth Request property values
    private static final String CLIENT_CREDENTIALS = "client_credentials";
    private static final String EXAMPLE_CLIENT_ID = "XXXXXXXXXXXXXXXXXXXXXXXXXXXX";
    private static final String EXAMPLE_CLIENT_SECRET = "XXXXXXXXXXXXXXXXXXXXXXXXXXXX";

    private static final String ACCESS_TOKEN = "access_token";

    private static final String REASON_HEADER = "X-Reason";

    // Id of the Cart we are expiring
    private static final String EXAMPLE_CART_ID = "7255c31c-39ac-4611-a2b7-974b0ebfa555";
```

```

public static void main(String[] args) {
    Client client = ClientBuilder.newClient();

    // To be able to access Orders Rest API, you need an access token.
    // First, we build the request data needed to gain an access token
    Form requestData = new Form();
    requestData.param(GRANT_TYPE, CLIENT_CREDENTIALS);
    requestData.param(CLIENT_ID, EXAMPLE_CLIENT_ID);
    requestData.param(CLIENT_SECRET, EXAMPLE_CLIENT_SECRET);

    // Then build and send the request
    Response oauthResponse = client.target(BRONTO_AUTH_PATH)
        .request(MediaType.APPLICATION_JSON)
        .accept(MediaType.TEXT_PLAIN_TYPE)
        .post(Entity.form(requestData));

    if (oauthResponse.getStatus() != Response.Status.OK.getStatusCode()) {
        throw new RuntimeException("Unable to get access token.");
    }

    // Retrieve the access token from the response
    Map<String, Object> responseData = oauthResponse.readEntity(Map.class);
    UUID accessToken = UUID.fromString((String)
    responseData.get(ACCESS_TOKEN));

    // Now to expire the Cart
    Response cartResponse = client.target(BRONTO_HOST)
        .path(EXPIRE_CART_PATH)
        .resolveTemplate(CART_ID, EXAMPLE_CART_ID)
        .request(MediaType.APPLICATION_JSON)
        .header("Authorization", "Bearer " + accessToken.toString())
        .post(Entity.json("")); // No body needed

    if (cartResponse.getStatus() !=
    Response.Status.NO_CONTENT.getStatusCode()) {
        String reason = cartResponse.getHeaderString(REASON_HEADER);
        throw new RuntimeException("Unable to expire Cart. Reason=" + reason);
    }
}
}

```

Fiddle Cart

Fiddle a cart with the given id.

Overview

When a cart is fiddled, the Bronto Marketing Platform is notified that the customer's cart is still ACTIVE and will reset the cart activity timer used by Bronto's cart abandonment feature. If the cart had previously been abandoned and the customer returns to the website, the fiddle will reset the cart's status to ACTIVE.

This request requires an access token with "orders/carts-write" scope. If the request is successful, a 204 No Content response will be returned. If no Cart can be found with the given cartId, a 404 Not Found response will be returned. If the Cart's status is EXPIRED or COMPLETE, a 409 Conflict response will be returned.

URI

```
PUT: https://rest.bronto.com/carts/{cartId}/fiddle
```

Parameters

Parameter	Type	Description
cartId	Required String	The unique ID of the cart.

Java Code Example

```
import java.util.Map;
import java.util.UUID;

import javax.ws.rs.client.Client;
import javax.ws.rs.client.ClientBuilder;
import javax.ws.rs.client.Entity;
import javax.ws.rs.core.Form;
import javax.ws.rs.core.MediaType;
import javax.ws.rs.core.Response;

public class FiddleCartExample {

    // Host
    private static final String BRONTO_HOST = "http://rest.bronto.com";
    private static final String BRONTO_AUTH_PATH = "https://auth.bronto.com/
oauth2/token";

    // Paths
    private static final String FIDDLE_CART_PATH = "carts/{cartId}/fiddle";
    private static final String CART_ID = "cartId";

    // OAuth Request property names
    private static final String GRANT_TYPE = "grant_type";
    private static final String CLIENT_ID = "client_id";
    private static final String CLIENT_SECRET = "client_secret";

    // OAuth Request property values
    private static final String CLIENT_CREDENTIALS = "client_credentials";
    private static final String EXAMPLE_CLIENT_ID = "XXXXXXXXXXXXXXXXXXXXXXXXX";
    private static final String EXAMPLE_CLIENT_SECRET
= "XXXXXXXXXXXXXXXXXXXXXXXXX";

    private static final String ACCESS_TOKEN = "access_token";

    private static final String REASON_HEADER = "X-Reason";

    // Id of the Cart we are fiddling
    private static final String EXAMPLE_CART_ID = "7255c31c-39ac-4611-
a2b7-974b0ebfa555";

    public static void main(String[] args) {
        Client client = ClientBuilder.newClient();

        // To be able to access Orders Rest API, you need an access token.
        // First, we build the request data needed to gain an access token
        Form requestData = new Form();
        requestData.param(GRANT_TYPE, CLIENT_CREDENTIALS);
        requestData.param(CLIENT_ID, EXAMPLE_CLIENT_ID);
        requestData.param(CLIENT_SECRET, EXAMPLE_CLIENT_SECRET);

        // Then build and send the request
        Response oauthResponse = client.target(BRONTO_AUTH_PATH)
```



```

        .request(MediaType.APPLICATION_JSON)
        .accept(MediaType.TEXT_PLAIN_TYPE)
        .post(Entity.form(requestData));

    if (oauthResponse.getStatus() != Response.Status.OK.getStatusCode()) {
        throw new RuntimeException("Unable to get access token.");
    }

    // Retrieve the access token from the response
    Map<String, Object> responseData = oauthResponse.readEntity(Map.class);
    UUID accessToken = UUID.fromString((String)
    responseData.get(ACCESS_TOKEN));

    // Now to fiddle the Cart
    Response cartResponse = client.target(BRONTO_HOST)
        .path(FIDDLE_CART_PATH)
        .resolveTemplate(CART_ID, EXAMPLE_CART_ID)
        .request(MediaType.APPLICATION_JSON)
        .header("Authorization", "Bearer " + accessToken.toString())
        .put(Entity.json("")); // No body needed

    if (cartResponse.getStatus() !=
    Response.Status.NO_CONTENT.getStatusCode()) {
        String reason = cartResponse.getHeaderString(REASON_HEADER);
        throw new RuntimeException("Unable to fiddle Cart. Reason=" + reason);
    }
}
}

```

Get Cart

View the contents of a cart using a cart's ID.

Overview

This request requires an access token with “orders/carts-write” scope. If the request is successful, a 201 Created response will be returned with the newly created Cart in the response body. If a Cart with the same customerCartId already exists, a 409 Conflict response will be returned.

URI

```

GET: https://rest.bronto.com/carts/{cartId}?
deliveryId={deliveryId}&messageId={messageId}&contactId={contactId}

```

Parameters

Parameter	Type	Description
cartId	Required String	The unique identifier of the cart.
deliveryId	Optional String	The unique identifier for a single delivery of a message associated with the order. You can use readDeliveries to return deliveryId as part of the deliveryObject .

Parameter	Type	Description
messageId	Optional String	The unique identifier for a single message associated with the order. You can use readMessages to return messageId as part of the messageObject .
contactId	Optional String	The unique identifier for a contact associated with the order. You can use readContacts or readContactsWithLatestUnsubscribeDate to return contactId as part of the contactObject .

Response

```
{
  url:string
  emailAddress:validly formatted email address
  status:ACTIVE | ABANDONED | EXPIRED | COMPLETE
  phase:SHOPPING | BILLING | PAYMENT | SHIPPING_INFO | SHIPPING_METHOD |
ORDER_REVIEW | ORDER_COMPLETE
  hasTracking:boolean
  customerCartId:string
  trackingCookieName:string
  trackingCookieValue:string
  discountAmount:number
  grandTotal:number
  lineItems:[
    {
      name:string
      other:string
      sku:string
      category:string
      imageUrl:string
      productUrl:string
      quantity:number
      salePrice:number
      totalPrice:number
      unitPrice:number
      description:string
      position:number
    }
  ]
  originIp: IPv4 or IPv6 address
  originUserAgent:string
  shippingAmount:number
  shippingDate: ISO-8601 datetime
  shippingDetails:string
  shippingTrackingUrl:string
  subtotal:number
  taxAmount:number
  cartId:UUID
  createdAt:ISO-8601 datetime
  updatedAt:ISO-8601 datetime
  currency:ISO-4217 currency code
}
```

Java Code Example

```

import java.util.Map;
import java.util.UUID;

import javax.ws.rs.client.Client;
import javax.ws.rs.client.ClientBuilder;
import javax.ws.rs.client.Entity;
import javax.ws.rs.core.Form;
import javax.ws.rs.core.MediaType;
import javax.ws.rs.core.Response;

public class GetCartExample {

    // Host
    private static final String BRONTO_HOST = "http://rest.bronto.com";
    private static final String BRONTO_AUTH_PATH = "https://auth.bronto.com/
oauth2/token";

    // Paths
    private static final String GET_CART_PATH = "carts/{cartId}";
    private static final String CART_ID = "cartId";

    // OAuth Request property names
    private static final String GRANT_TYPE = "grant_type";
    private static final String CLIENT_ID = "client_id";
    private static final String CLIENT_SECRET = "client_secret";

    // OAuth Request property values
    private static final String CLIENT_CREDENTIALS = "client_credentials";
    private static final String EXAMPLE_CLIENT_ID = "xxxxxxxxxxxxxxxxxxxxxxxx";
    private static final String EXAMPLE_CLIENT_SECRET
= "xxxxxxxxxxxxxxxxxxxxxxxx";

    private static final String ACCESS_TOKEN = "access_token";

    private static final String REASON_HEADER = "X-Reason";

    // Id of the Cart we are getting
    private static final String EXAMPLE_CART_ID = "7255c31c-39ac-4611-
a2b7-974b0ebfa555";

    public static void main(String[] args) {
        Client client = ClientBuilder.newClient();

        // To be able to access Orders Rest API, you need an access token.
        // First, we build the request data needed to gain an access token
        Form requestData = new Form();
        requestData.param(GRANT_TYPE, CLIENT_CREDENTIALS);
        requestData.param(CLIENT_ID, EXAMPLE_CLIENT_ID);
        requestData.param(CLIENT_SECRET, EXAMPLE_CLIENT_SECRET);

        // Then build and send the request
        Response oauthResponse = client.target(BRONTO_AUTH_PATH)
            .request(MediaType.APPLICATION_JSON)
            .accept(MediaType.TEXT_PLAIN_TYPE)
            .post(Entity.form(requestData));

        if (oauthResponse.getStatus() != Response.Status.OK.getStatusCode()) {
            throw new RuntimeException("Unable to get access token.");
        }
    }
}

```

```

    }

    // Retrieve the access token from the response
    Map<String, Object> responseData = oauthResponse.readEntity(Map.class);
    UUID accessToken = UUID.fromString((String)
responseData.get(ACCESS_TOKEN));

    // Now to get the Cart
    Response cartResponse = client.target(BRONTO_HOST)
        .path(GET_CART_PATH)
        .resolveTemplate(CART_ID, EXAMPLE_CART_ID)
        .request(MediaType.APPLICATION_JSON)
        .header("Authorization", "Bearer " + accessToken.toString())
        .get();

    if (cartResponse.getStatus() ==
Response.Status.NOT_FOUND.getStatusCode()) {
        throw new RuntimeException("No Cart found with cartId=" +
EXAMPLE_CART_ID);
    } else if (cartResponse.getStatus() !=
Response.Status.OK.getStatusCode()) {
        String reason = cartResponse.getHeaderString(REASON_HEADER);
        throw new RuntimeException("Unable to get Cart. Reason=" + reason);
    }

    // Retrieve the Cart from the response
    Map<String, Object> cart = cartResponse.readEntity(Map.class);
    System.out.println(cart);
}
}

```

Search For Carts

Search for carts using a customerCartId.

Overview

This request requires an access token with “orders/carts-read” scope. If the request is successful a Cart is found, a 200 Ok response will be returned including the Cart in the response body. If no results are found, a 204 No Content response will be returned. If there is a problem with any of the query parameters, a 400 Bad Request response will be returned with an explanation of the error in the X-Reason header.

URI

```

GET: https://rest.bronto.com/carts/customerCartId/{customerCartId}?
deliveryId={deliveryId}&messageId={messageId}&contactId={contactId}

```

Parameters

Parameter	Type	Description
customerCartId	Required String	Customer’s ID that is associated with the cart
deliveryId	Optional String	The unique identifier for a single delivery of a message associated with the cart. You can use readDeliveries to return deliveryId as part of the deliveryObject .

Parameter	Type	Description
messageId	Optional String	The unique identifier for a single message associated with the cart. You can use readMessages to return messageId as part of the messageObject .
contactId	Optional String	The unique identifier for a contact associated with the cart. You can use readContacts or readContactsWithLatestUnsubscribeDate to return contactId as part of the contactObject .

Response

A response containing a list of the orders found, if any.

```
[
  {
    url:string
    emailAddress:string
    status:ACTIVE | ABANDONED | EXPIRED | COMPLETE
    phase:SHOPPING | BILLING | PAYMENT | SHIPPING_INFO | SHIPPING_METHOD |
ORDER_REVIEW | ORDER_COMPLETE
    hasTracking:boolean
    customerCartId:string
    trackingCookieName:string
    trackingCookieValue:string
    discountAmount:number
    grandTotal:number
    lineItems:[
      {
        name:string
        other:string
        sku:string
        category:string
        imageUrl:string
        productUrl:string
        quantity:number
        salePrice:number
        totalPrice:number
        unitPrice:number
        description:string
        position:number
      }
    ]
    originIp:IPv4 or IPv6 address
    originUserAgent:string
    shippingAmount:number
    shippingDate:ISO-8601 datetime
    shippingDetails:string
    shippingTrackingUrl:string
    subtotal:number
    taxAmount:number
    cartId:UUID
    createdAt:ISO-8601 datetime
    updatedAt:ISO-8601 datetime
    currency:ISO-4217 currency code
  }
]
```

]

Java Code Example

```

import java.util.Map;
import java.util.UUID;

import javax.ws.rs.client.Client;
import javax.ws.rs.client.ClientBuilder;
import javax.ws.rs.client.Entity;
import javax.ws.rs.core.Form;
import javax.ws.rs.core.MediaType;
import javax.ws.rs.core.Response;

public class SearchForCartExample {

    // Host
    private static final String BRONTO_HOST = "http://rest.bronto.com";
    private static final String BRONTO_AUTH_PATH = "https://auth.bronto.com/
oauth2/token";

    // Paths
    private static final String GET_CART_BY_CUSTOMER_ID_PATH = "carts/
customerCartId/{customerCartId}";
    private static final String CUSTOMER_CART_ID = "customerCartId";

    // OAuth Request property names
    private static final String GRANT_TYPE = "grant_type";
    private static final String CLIENT_ID = "client_id";
    private static final String CLIENT_SECRET = "client_secret";

    // OAuth Request property values
    private static final String CLIENT_CREDENTIALS = "client_credentials";
    private static final String EXAMPLE_CLIENT_ID = "XXXXXXXXXXXXXXXXXXXXXXXXX";
    private static final String EXAMPLE_CLIENT_SECRET
= "XXXXXXXXXXXXXXXXXXXXXXXXX";

    private static final String ACCESS_TOKEN = "access_token";

    private static final String REASON_HEADER = "X-Reason";

    // Customer's Id of the Cart we are getting
    private static final String EXAMPLE_CART_ID = "abc-123";

    public static void main(String[] args) {

        Client client = ClientBuilder.newClient();

        // To be able to access Orders Rest API, you need an access token.
        // First, we build the request data needed to gain an access token
        Form requestData = new Form();
        requestData.param(GRANT_TYPE, CLIENT_CREDENTIALS);
        requestData.param(CLIENT_ID, EXAMPLE_CLIENT_ID);
        requestData.param(CLIENT_SECRET, EXAMPLE_CLIENT_SECRET);

        // Then build and send the request
        Response oauthResponse = client.target(BRONTO_AUTH_PATH)
            .request(MediaType.APPLICATION_JSON)
            .accept(MediaType.TEXT_PLAIN_TYPE)

```

```

        .post(Entity.form(requestData));

    if (oauthResponse.getStatus() != Response.Status.OK.getStatusCode()) {
        throw new RuntimeException("Unable to get access token.");
    }

    // Retrieve the access token from the response
    Map<String, Object> responseData = oauthResponse.readEntity(Map.class);
    UUID accessToken = UUID.fromString((String)
    responseData.get(ACCESS_TOKEN));

    // Now to get the Cart
    Response cartResponse = client.target(BRONTO_HOST)
        .path(GET_CART_BY_CUSTOMER_ID_PATH)
        .resolveTemplate(CUSTOMER_CART_ID, EXAMPLE_CART_ID)
        .request(MediaType.APPLICATION_JSON)
        .header("Authorization", "Bearer " + accessToken.toString())
        .get();

    if (cartResponse.getStatus() ==
    Response.Status.NOT_FOUND.getStatusCode()) {
        throw new RuntimeException("No Cart found with cartId=" +
    EXAMPLE_CART_ID);
    } else if (cartResponse.getStatus() !=
    Response.Status.OK.getStatusCode()) {
        String reason = cartResponse.getHeaderString(REASON_HEADER);
        throw new RuntimeException("Unable to get Cart. Reason=" + reason);
    }

    // Retrieve the Cart from the response
    Map<String, Object> cart = cartResponse.readEntity(Map.class);
    System.out.println(cart);
}
}

```

Update Cart

Make an update to an existing cart. Properties that are not included in the request body will remain unchanged on the existing cart. No properties are required in the update.

Overview

You can provide tracking attribution by including a tid with your request or a TrackingCookieName and TrackingCookieValue pair. For more information see Orders REST API Tracking Attribution.

This request requires an access token with “orders/carts-write” scope. If the request is successful, a 200 Ok response will be returned with the latest version of the Cart in the response body. If the Cart’s status is either EXPIRED or COMPLETE, a 409 Conflict response will be returned. If there is a problem with any of the data in the request body, a 400 Bad Request response will be returned with an explanation of the error in the X-Reason header.

URI

```

POST: https://rest.bronto.com/carts/{cartId}?
createContact={boolean}&ignoreInvalidTid={boolean}

```

Parameters

Parameter	Type	Description
cartId	Required String	The ID for the cart.

Parameter	Type	Description
createContact	Optional Boolean	Determines behavior when the request contains an email address that is not associated with a Contact. If true, a transactional Contact will be created. If false, the service will return a 409 Conflict response. The default value is false.
ignoreInvalidTid	Optional Boolean	Determines behavior when the request contains an invalid TID or TrackingCookieName and TrackingCookieValue pair. If true, the service will ignore the invalid values and continue.

Request Body

The cart data used to update the cart.

```
{
  url:string
  discountAmount:number
  emailAddress:validly formatted email address
  contactId:string
  grandTotal:number
  deliveryId:string
  lineItems:[
    {
      name:string
      description:string
      sku:string
      other:string
      imageUrl:string
      category:string
      productUrl:string
      quantity:number
      salePrice:number
      totalPrice:number
      unitPrice:number
    }
  ]
  originIp:IPv4 or IPv6 address
  messageId:string
  originUserAgent:string
  shippingAmount:number
  shippingDate:ISO-8601 datetime
  shippingDetails:string
  shippingTrackingUrl:string
  subtotal:number
  taxAmount:number
  trackingCookieName:string
  trackingCookieValue:stringtid:string
  tid:string
  phase:SHOPPING | BILLING | PAYMENT | SHIPPING_INFO | SHIPPING_METHOD |
ORDER_REVIEW | ORDER_COMPLETE
  currency:ISO-4217 currency code
}
```


Response

A response containing the updated cart.

```
{
  url:string
  emailAddress:string
  contactId:string
  status:ACTIVE | ABANDONED | EXPIRED | COMPLETE
  phase:SHOPPING | BILLING | PAYMENT | SHIPPING_INFO | SHIPPING_METHOD |
ORDER_REVIEW | ORDER_COMPLETE
  hasTracking:boolean
  customerCartId:string
  trackingCookieName:string
  trackingCookieValue:string
  deliveryId:string
  discountAmount:number
  grandTotal:number
  lineItems:[
    {
      name:string
      other:string
      sku:string
      category:string
      imageUrl:string
      productUrl:string
      quantity:number
      salePrice:number
      totalPrice:number
      unitPrice:number
      description:string
      position:number
    }
  ]
  originIp:IPv4 or IPv6 address
  messageId:string
  originUserAgent:string
  shippingAmount:number
  shippingDate:ISO-8601 datetime
  shippingDetails:string
  shippingTrackingUrl:string
  subtotal:number
  taxAmount:number
  cartId:UUID
  createdAt:ISO-8601 datetime
  updatedAt:ISO-8601 datetime
  currency:ISO-4217 currency code
}
```

Java Code Example

```
import java.util.Arrays;
import java.util.HashMap;
import java.util.Map;
import java.util.UUID;

import javax.ws.rs.client.Client;
import javax.ws.rs.client.ClientBuilder;
import javax.ws.rs.client.Entity;
import javax.ws.rs.core.Form;
import javax.ws.rs.core.MediaType;
import javax.ws.rs.core.Response;
```

```

public class UpdateCartExample {

    // Host
    private static final String BRONTO_HOST = "http://rest.bronto.com";
    private static final String BRONTO_AUTH_PATH = "https://auth.bronto.com/
oauth2/token";

    // Paths
    private static final String UPDATE_CART_PATH = "carts/{cartId}";
    private static final String CART_ID = "cartId";
    private static final String CREATE_CONTACT = "createContact";
    private static final String IGNORE_INVALID_TRACKING
= "ignoreInvalidTracking";

    // OAuth Request property names
    private static final String GRANT_TYPE = "grant_type";
    private static final String CLIENT_ID = "client_id";
    private static final String CLIENT_SECRET = "client_secret";

    // OAuth Request property values
    private static final String CLIENT_CREDENTIALS = "client_credentials";
    private static final String EXAMPLE_CLIENT_ID = "xxxxxxxxxxxxxxxxxxxxxxxx";
    private static final String EXAMPLE_CLIENT_SECRET
= "xxxxxxxxxxxxxxxxxxxxxxxx";

    private static final String ACCESS_TOKEN = "access_token";

    private static final String REASON_HEADER = "X-Reason";

    // Id of the Cart we are updating
    private static final String EXAMPLE_CART_ID = "7255c31c-39ac-4611-
a2b7-974b0ebfa555";

    public static void main(String[] args) {
        Client client = ClientBuilder.newClient();

        // To be able to access Orders Rest API, you need an access token.
        // First, we build the request data needed to gain an access token
        Form requestData = new Form();
        requestData.param(GRANT_TYPE, CLIENT_CREDENTIALS);
        requestData.param(CLIENT_ID, EXAMPLE_CLIENT_ID);
        requestData.param(CLIENT_SECRET, EXAMPLE_CLIENT_SECRET);

        // Then build and send the request
        Response oauthResponse = client.target(BRONTO_AUTH_PATH)
            .request(MediaType.APPLICATION_JSON)
            .accept(MediaType.TEXT_PLAIN_TYPE)
            .post(Entity.form(requestData));

        if (oauthResponse.getStatus() != Response.Status.OK.getStatusCode()) {
            throw new RuntimeException("Unable to get access token.");
        }

        // Retrieve the access token from the response
        Map<String, Object> responseData = oauthResponse.readEntity(Map.class);
        UUID accessToken = UUID.fromString((String)
responseData.get(ACCESS_TOKEN));

        // Now to update the Cart, we first build the request data
        Map<String, Object> lineItemData = new HashMap<String, Object>(2);

```

```

lineItemData.put("name", "Coffee Mug");
lineItemData.put("description", "A cool coffee mug");
lineItemData.put("category", "Kitchen Stuffs");
lineItemData.put("other", "5oz");
lineItemData.put("imageUrl", "http://www.example.com/
images/0003105.jpg");
lineItemData.put("productUrl", "http://www.example.com/
products/0003105");
lineItemData.put("sku", "0003105");
lineItemData.put("quantity", 1);
lineItemData.put("totalPrice", 3.99f);
lineItemData.put("unitPrice", 3.99f);
lineItemData.put("salePrice", 3.99f);

Map<String, Object> cartData = new HashMap<String, Object>(2);
cartData.put("emailAddress", "example@email.com");
cartData.put("currency", "USD");
cartData.put("grandTotal", 4.27f);
cartData.put("discountAmount", 0);
cartData.put("taxAmount", 0.28f);
cartData.put("subtotal", 3.99f);
cartData.put("lineItems", Arrays.asList(lineItemData));
cartData.put("phase", "SHOPPING");
cartData.put("url", "http://www.example.com/carts/abc-123");
cartData.put("originIp", "127.0.0.1");
cartData.put("originUserAgent",
    "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_9_4)
AppleWebKit/537.36 (KHTML, like Gecko) Chrome/46.0.2490.71 Safari/537.36");

cartData.put("trackingCookieName", "tid_bkycggbjquivddbiddxylaufhfdbkj");

cartData.put("trackingCookieValue", "3.Ag.AQ.AQ.AQ.AQ.t..l.AQ.n.VO4AkA.VO4AkA.G89Jov");
cartData.put("shippingAmount", 0);
cartData.put("shippingDate", "2015-01-02");
cartData.put("shippingDetails", "Free next day shipping");
cartData.put("shippingTrackingUrl", "http://www.shipping.com/1234");

// Update the Cart
Response cartResponse = client.target(BRONTO_HOST)
    .path(UPDATE_CART_PATH)
    .resolveTemplate(CART_ID, EXAMPLE_CART_ID)
    .queryParam(CREATE_CONTACT, true)
    .queryParam(IGNORE_INVALID_TRACKING, true)
    .request(MediaType.APPLICATION_JSON)
    .header("Authorization", "Bearer " + accessToken.toString())
    .post(Entity.json(cartData));

if (cartResponse.getStatus() ==
Response.Status.NOT_FOUND.getStatusCode()) {
    throw new RuntimeException("No Cart found with cartId=" +
EXAMPLE_CART_ID);
} else if (cartResponse.getStatus() !=
Response.Status.OK.getStatusCode()) {
    String reason = cartResponse.getHeaderString(REASON_HEADER);
    throw new RuntimeException("Unable to update Cart. Reason=" + reason);
}

// Retrieve the Cart from the response
Map<String, Object> cart = cartResponse.readEntity(Map.class);
System.out.println(cart);
}

```

```
}
```

Orders REST API Tracking Attribution

You can pass tracking data to Bronto using a tid. A tid is a unique id that associates an order with a specific contact and delivery. How you can identify the tid is dependent upon whether or not your account uses a private domain.

To determine this:

1. Log into Bronto.

If you cannot log into Bronto ask the person who uses your company's login to complete these steps.

2. Go to **Home > Settings > Commerce**.
3. In the **Order Settings** section, check to see if **Add tracking URL parameters used to create and use first-party cookies** is checked.
4. If this:
 - is checked, your account does not use a private domain. Continue to the When You Use Bronto's Domain section.
 - is NOT checked your account uses a private domain. Continue to the When You Use a Private Domain section.

When You Use Bronto's Domain

In this case, tracking information is passed as part of a query string request to the commerce site. To find the tid, parse the query string for the parameter `_bta_tid`. For example: A customer clicks on an email link and is directed to the commerce site:

```
https://shopping.example.com?
_bta_tid=12345678901234567890123456789012345678901234567890
```

Then you capture the value from `_bta_tid`, store it so it is associated with the shopping session, and attach it to cart or order REST requests. In this example, the tid element is

```
{ "tid": "12345678901234567890123456789012345678901234567890"; }.
```

When You Use a Private Domain

In this case, Bronto sets a cookie on the domain of your commerce site that contains the tid. This cookie is named `tid_{encrypted site_id}` so you will need to determine what your encrypted site ID is in order to find the cookie. The encrypted site ID is the `bsw.src` value found in the Bronto embed code. You can either read the embed code from where it is on your site, or log into Bronto, go to **Home > Settings > Commerce** and read the embed code in the **Tag Manager** section.

Once you have your encrypted site ID, open the file named `tid_{encrypted site_id}`. The value string is your tid. For example: A customer clicks on a link and is directed to the commerce site `https://shopping.example.com` Bronto sets a cookie:

```
name = tid_3gvfksuhdbvbicfddvov4g3i4xmfitkgvdsd6720yko4vtp1ngyzz
value = 12345678901234567890123456789012345678901234567890
```

Then you capture the value from the cookie, store it so it is associated with the shopping session, and attach it to cart or order REST requests. In this example, the tid element is:

```
{ "tid": "12345678901234567890123456789012345678901234567890"; }.
```

REST Product Service

The Product Service offers a REST API for pushing in full product catalog feeds and individual product updates, as well as retrieving products. It is accessible using HTTPS and secured with OAuth 2.0.

Get Product

The products GET function where an individual product id is included returns the complete set of data stored in Bronto for the product matching the given product id in the function call.

Overview

If it is not found, a 404 error code is returned.

URI

```
GET: https://rest.bronto.com/products/public/catalogs/{catalogId}/products/{productId}
```

Parameters

Parameter	Type	Description
fieldNames	Optional	You can filter the data that is returned by specifying which product fields you want to get information about. The list of valid fields can be found on the Product Fields page in the BMP. If you do not include any fieldNames then all of the data for the product is returned. For example: /products/public/catalogs/{catalogId}/products/{productId}

Response

```
{
  id:string,
  fields:[
    {
      name:string,
      value:object,
      id:number,
      type:string
    }
  ]
}
```

Python Code Example

```
from __future__ import print_function

# Requests documentation: http://docs.python-requests.org/
import requests

# API authorization
bronto_access_token = "a valid REST API access token"
session = requests.Session()
session.headers["Authorization"] = "Bearer " + bronto_access_token

catalog_id = your catalog ID here

# get a single product with all of its fields
product_id_1 = "product-0001"
```

```

url_fmt = "https://rest.bronto.com/products/public/catalogs/{catalog_id}/
products/{product_id}"
url = url_fmt.format(catalog_id=catalog_id, product_id=product_id_1)

response = session.get(url)

if response.ok:
    product = response.json()
    print("Product with all fields:", product)
else:
    print("Could not get product " + product_id_1 + " (status code={},
content='{}', headers={})".format(
        response.status_code, response.text, response.headers
    ))

# get a single product but only some of its fields
url += "?fieldNames=price&fieldNames=quantity"

response = session.get(url)

if response.ok:
    product = response.json()
    print("Product with only price and quantity fields:", product)
else:
    print("Could not get product " + product_id_1 + " (status code={},
content='{}', headers={})".format(
        response.status_code, response.text, response.headers
    ))

```

Java Code Example

```

import javax.ws.rs.client.Client;
import javax.ws.rs.client.ClientBuilder;
import javax.ws.rs.client.Entity;
import javax.ws.rs.core.Form;
import javax.ws.rs.core.MediaType;
import javax.ws.rs.core.Response;
import javax.ws.rs.core.UriBuilder;
import java.net.URI;
import java.util.Arrays;
import java.util.List;
import java.util.Map;

public class GetProduct {

    // - Do not change
    -----

    private static final String BRONTO_REST_HOST = "https://
rest.bronto.com";
    private static final String RELATIVE_ENDPOINT_PATH = "products/public/
catalogs/{catalogId}/products/{productId}";

    private static final String BRONTO_AUTH_PATH = "https://auth.bronto.com/
oauth2/token";

    //-----

    // Modify the below values for your specific site and catalog

```

```

// Replace the below OAuth Request property values with your site
specific credentials
private static final String CLIENT_ID = "1234567890";
private static final String CLIENT_SECRET = "0987654321";

// Replace with your catalog ID
private static final String CATALOG_ID = "1";

// Replace with your product ID
private static final String PRODUCT_ID = "test_id_123";

//OPTIONAL: Field name parameters limit the fields returned in the call.
// If you pass in no field name parameters, ALL fields will be returned.
private static final List<String> FIELDS =
Arrays.asList("title", "description");

public static void main(String[] args) {

    Client client = ClientBuilder.newClient();

    // To be able to access Any Bronto Rest API, you need an access
token.
    // First, we build the request data needed to gain an access token
    Form requestData = new Form();
    requestData.param("grant_type", "client_credentials");
    requestData.param("client_id", CLIENT_ID);
    requestData.param("client_secret", CLIENT_SECRET);

    // Then build and send the request
    Response oauthResponse = client.target(BRONTO_AUTH_PATH)
        .request(MediaType.APPLICATION_JSON)
        .accept(MediaType.TEXT_PLAIN_TYPE)
        .post(Entity.form(requestData));

    if (oauthResponse.getStatus() != Response.Status.OK.getStatusCode())
    {
        throw new RuntimeException("Unable to get access token.");
    }

    // Retrieve the access token from the response
    Map<String, Object> responseData =
oauthResponse.readEntity(Map.class);
    String accessToken = (String) responseData.get("access_token");

    // Here begins the Get Product code
    -----

    // Build the URL
    UriBuilder builder = UriBuilder.fromUri(BRONTO_REST_HOST)
        .path(RELATIVE_ENDPOINT_PATH)
        .resolveTemplate("catalogId", CATALOG_ID)
        .resolveTemplate("productId", PRODUCT_ID);

    // Add optional field name query parameters
    for (String field : FIELDS) {
        builder.queryParam("fieldNames", field);
    }

    URI uri = builder.build();

    //Create and execute the authorized get request
    Response productResponse = client.target(uri)
        .request(MediaType.APPLICATION_JSON)

```

```

        .header("Authorization", "Bearer " + accessToken)
        .get();

    // Retrieve the Product from the response
    String product = productResponse.readEntity(String.class);
    System.out.println(product);
}
}

```

Get Products

This returns the complete set of data stored in Bronto for each product matching the given Product IDs in the call.

Overview

If none of the listed products are found, a 404 error code is returned.

URI

GET: <https://rest.bronto.com/products/public/catalogs/{catalogId}/products>

Parameters

Parameter	Type	Description
productIds	Required	A list of the Product IDs you want to get data for. For example: <code>/products/public/products?productIds=X&productIds=X</code>
fieldNames	Optional	<p>You can filter the data that is returned by specifying which product fields you want to get information about. The list of valid fields can be found on the Product Fields page in the BMP. If you do not include any fieldNames then all of the data for the product is returned. For example:</p> <pre> /products/public/ catalogs/{catalogId}/ products </pre>

Response

```

[
  {
    id:string,
    fields:[
      {
        name:string,
        value:object,
        id:number,
        type:string
      }
    ]
  }, {

```



```

    id:string,
    fields:[
      {
        name:string,
        value:object,
        id:number,
        type:string
      }
    ]
  }
]

```

Python Code Example

```

from __future__ import print_function

# Requests documentation: http://docs.python-requests.org/
import requests

# API authorization
bronto_access_token = "a valid REST API access token"
session = requests.Session()
session.headers["Authorization"] = "Bearer " + bronto_access_token

catalog_id = your catalog ID here

# get multiple products with all of their fields
product_id_1 = "product-0001"
product_id_2 = "product-0002"

url_fmt = "https://rest.bronto.com/products/public/catalogs/{catalog_id}/products"
url = url_fmt.format(catalog_id=catalog_id)
url += "?productId=" + product_id_1 + "&productId=" + product_id_2

response = session.get(url)

if response.ok:
    products = response.json()
    print("Products with all fields:", products)
else:
    print("Could not get products (status code={}, content='{}', headers={})".format(
        response.status_code, response.text, response.headers
    ))

# get multiple products with only some of their fields
url += "&fieldNames=price&fieldNames=quantity"

response = session.get(url)

if response.ok:
    products = response.json()
    print("Products with only price and quantity fields:", products)
else:
    print("Could not get products (status code={}, content='{}', headers={})".format(
        response.status_code, response.text, response.headers
    ))

```

```
))
```

Java Code Example

```
import javax.ws.rs.client.Client;
import javax.ws.rs.client.ClientBuilder;
import javax.ws.rs.client.Entity;
import javax.ws.rs.core.Form;
import javax.ws.rs.core.MediaType;
import javax.ws.rs.core.Response;
import javax.ws.rs.core.UriBuilder;
import java.net.URI;
import java.util.Arrays;
import java.util.List;
import java.util.Map;

public class GetProducts {

    // - Do not change
    -----

    private static final String BRONTO_REST_HOST = "https://
rest.bronto.com";
    private static final String RELATIVE_ENDPOINT_PATH = "products/public/
catalogs/{catalogId}/products";

    private static final String BRONTO_AUTH_PATH = "https://auth.bronto.com/
oauth2/token";

    //-----

    // Modify the below values for your specific site and catalog

    // Replace the below OAuth Request property values with your site
specific credentials
    private static final String CLIENT_ID = "1234567890";
    private static final String CLIENT_SECRET = "0987654321";

    // Replace with your catalog ID
    private static final String CATALOG_ID = "1";

    // Replace with your product IDs
    private static final List<String> PRODUCT_IDS =
Arrays.asList("test_id_1", "test_id_2");

    //OPTIONAL: Field name parameters limit the fields returned in the call.
    // If you pass in no field name parameters, ALL fields will be returned.
    private static final List<String> FIELDS =
Arrays.asList("title", "description");

    public static void main(String[] args) {

        Client client = ClientBuilder.newClient();

        // To be able to access Any Bronto Rest API, you need an access
token.
        // First, we build the request data needed to gain an access token
        Form requestData = new Form();
        requestData.param("grant_type", "client_credentials");
        requestData.param("client_id", CLIENT_ID);
        requestData.param("client_secret", CLIENT_SECRET);
```

```

// Then build and send the request
Response oauthResponse = client.target(BRONTO_AUTH_PATH)
    .request(MediaType.APPLICATION_JSON)
    .accept(MediaType.TEXT_PLAIN_TYPE)
    .post(Entity.form(requestData));

if (oauthResponse.getStatus() != Response.Status.OK.getStatusCode())
{
    throw new RuntimeException("Unable to get access token.");
}

// Retrieve the access token from the response
Map<String, Object> responseData =
oauthResponse.readEntity(Map.class);
String accessToken = (String) responseData.get("access_token");

// Here begins the Get Products code
-----

// Build the URL
UriBuilder builder = UriBuilder.fromUri(BRONTO_REST_HOST)
    .path(RELATIVE_ENDPOINT_PATH)
    .resolveTemplate("catalogId", CATALOG_ID);

// Add product id query parameters
for (String productId : PRODUCT_IDS) {
    builder.queryParam("productIds", productId);
}

// Add optional field name query parameters
for (String field : FIELDS) {
    builder.queryParam("fieldNames", field);
}

URI uri = builder.build();

//Create and execute the authorized get request
Response productResponse = client.target(uri)
    .request(MediaType.APPLICATION_JSON)
    .header("Authorization", "Bearer " + accessToken)
    .get();

// Retrieve the Product from the response
String product = productResponse.readEntity(String.class);
System.out.println(product);
}
}

```

Import Product Data Feed

The `feed_import` function allows you to import a new product feed into Bronto.

Overview

The feed file imported should represent your entire product data, as any products previously imported not submitted in this feed will be archived. This function is a multi-part form data endpoint that accepts incoming product feeds for asynchronous processing, and returns a transaction UUID. Any valid feed file that you can import via the Product Service inside of the BMP is acceptable input.



Important: Your product feed file cannot contain more than 3 million products.

URI

POST: https://rest.bronto.com/products/public/feed_import

Parameters

Parameter	Type	Description
catalogId	Required ID	Your catalog id. This ID can be found on the Products Overview page in the BMP, located on the bottom-right part of the page. It is titled "Products API ID".
feed	Required	The file that is being uploaded

Response

Transaction UUID

Python Code Example

```
from __future__ import print_function

# Requests documentation: http://docs.python-requests.org/
import requests

# API authorization
bronto_access_token = "a valid REST API access token"
headers = {"Authorization": "Bearer " + bronto_access_token}

# import feed
feed_filepath = "my_products_feed.csv"
catalog_id = "your catalog ID here"

with open(feed_filepath, "rb") as feed_file:
    files = {
        "feed": feed_file,
        "catalogId": str(catalog_id)
    }

    url = "https://rest.bronto.com/products/public/feed_import"
    response = requests.post(url, files=files, headers=headers)

# check response
if response.ok:
    print("New feed import ID is", response.text)
else:
    print("Could not import feed (status code={}, content='{}', headers={})".format(
        response.status_code, response.text, response.headers
    ))
```

Java Code Example

```
import org.glassfish.jersey.media.multipart.FormDataMultiPart;
import org.glassfish.jersey.media.multipart.MultiPartFeature;
```

```

import org.glassfish.jersey.media.multipart.file.FileDataBodyPart;

import javax.ws.rs.client.Client;
import javax.ws.rs.client.ClientBuilder;
import javax.ws.rs.client.Entity;
import javax.ws.rs.core.Form;
import javax.ws.rs.core.MediaType;
import javax.ws.rs.core.Response;
import javax.ws.rs.core.UriBuilder;
import java.io.File;
import java.io.IOException;
import java.net.URI;
import java.util.Map;

public class ImportFeed {

    // - Do not change
    -----

    private static final String BRONTO_REST_HOST = "https://
rest.bronto.com";
    private static final String RELATIVE_ENDPOINT_PATH = "products/public/
feed_import";

    private static final String BRONTO_AUTH_PATH = "https://auth.bronto.com/
oauth2/token";

    //-----

    // Modify the below values for your specific site and catalog

    // Replace the below OAuth Request property values with your site
specific credentials
    private static final String CLIENT_ID = "1234567890";
    private static final String CLIENT_SECRET = "0987654321";

    // Replace with your catalog ID
    private static final String CATALOG_ID = "1";

    // Replace with path to your feed file
    private static final String FEED_FILE_PATH = "/path/to/feed/file";

    public static void main(String[] args) {

        Client client = ClientBuilder.newClient();
        client.register(MultiPartFeature.class);

        // To be able to access Any Bronto Rest API, you need an access
token.
        // First, we build the request data needed to gain an access token
        Form requestData = new Form();
        requestData.param("grant_type", "client_credentials");
        requestData.param("client_id", CLIENT_ID);
        requestData.param("client_secret", CLIENT_SECRET);

        // Then build and send the request
        Response oauthResponse = client.target(BRONTO_AUTH_PATH)
            .request(MediaType.APPLICATION_JSON)
            .accept(MediaType.TEXT_PLAIN_TYPE)
            .post(Entity.form(requestData));

        if (oauthResponse.getStatus() != Response.Status.OK.getStatusCode())
    {

```

```

        throw new RuntimeException("Unable to get access token.");
    }

    // Retrieve the access token from the response
    Map<String, Object> responseData =
oauthResponse.readEntity(Map.class);
    String accessToken = (String) responseData.get("access_token");

    // Here begins the Import Product Feed code
    -----

    FormDataMultiPart formDataMultiPart = null;

    // Build the URL
    URI uri = UriBuilder.fromUri(BRONTO_REST_HOST)
        .path(RELATIVE_ENDPOINT_PATH)
        .build();

    try {
        formDataMultiPart = new FormDataMultiPart();
        formDataMultiPart.field("catalogId", CATALOG_ID);
        formDataMultiPart.bodyPart(new FileDataBodyPart("feed", new
File(FEED_FILE_PATH)));

        //Create and execute the authorized post request
        Response response = client.target(uri)
            .request(MediaType.MULTIPART_FORM_DATA_TYPE)
            .accept(MediaType.TEXT_PLAIN_TYPE)
            .header("Authorization", "Bearer " + accessToken)
            .post(Entity.entity(formDataMultiPart,
formDataMultiPart.getMediaType()));

        // Retrieve the Feed UUID and the returned status code from the
response
        String output = response.readEntity(String.class);
        System.out.println(output);
        System.out.println(response.getStatus());

    } finally {
        // Make sure you've closed everything
        if (formDataMultiPart != null) {
            try {
                formDataMultiPart.close();
            } catch (IOException e) {
                // Handle IOException
                System.out.println("Oops: " + e.getStackTrace());
            }
        }
    }
}
}
}

```

Update Products

Allows synchronous batch updates to the products in your catalog.

Overview

Accepts an array of product update entities, which must contain all the fields (field name : field value) in the product. Normal validation of the product field values will be applied, and individual products updates will be rejected if they fail validation. An array of String errors will be returned for each product that is not updated.



Note: You cannot add new products or archive old using this endpoint. Those updates must be done via a full feed import.

Results of these updates do not trigger email notifications or the creation of a new record in Bronto on the Product Import History page. You will however see changes represented in the Products Search results. To see errors that have occurred with specific product updates, you must view the responses to the API calls.

URI

PUT: <https://rest.bronto.com/products/public/catalogs/{catalogId}/products>

Request Body

```
[
  {
    "fields": {
      "product_id": "product-0002",
      "parent_product_id": None,
      "product_url": "http://mysite.com/products/0002",
      "title": "Product 0002 title",
      "description": "Product 0002 description",
      "price": "27.00 USD",
      "quantity": 25,
      "image_url": "http://mysite.com/products/0002.png",
      "additional_images": None,
      "rating": 3.9,
      "inventory_threshold": 5,
      "availability": "Available",
      "availability_date": "",
      "product_category": "Miscellaneous",
      "sale price effective date":
"2010-01-10T00:00-0100/2020-01-10T00:00-0100",
      "tax": "US:1111:8.75, US:1111:7.75",
      "gender": "Male",
      "color": "Green",
      "size": "M",
    }
  },
  {
    "fields": {
      "product_id": "product-0002",
      "parent_product_id": None,
      "product_url": "http://mysite.com/products/0002",
      "title": "Product 0002 title",
      "description": "Product 0002 description",
      "price": "27.00 USD",
      "quantity": 25,
      "image_url": "http://mysite.com/products/0002.png",
      "additional_images": None,
      "rating": 3.9,
      "inventory_threshold": 5,
      "availability": "Available",
      "availability_date": "",
      "product_category": "Miscellaneous",
      "sale price effective date":
"2010-01-10T00:00-0100/2020-01-10T00:00-0100",
      "tax": "US:1111:8.75, US:1111:7.75",
      "gender": "Male",
      "color": "Green",
      "size": "M",
    }
  }
]
```

```
}
]
```

Response

```
[
  "ERROR - id=1 - The following mapped fields were not present in the product:
    rating",
  "ERROR - id=5 - The following mapped fields were not present in the product:
    margin"
]
```

Python Code Example

```
from __future__ import print_function

# Requests documentation: http://docs.python-requests.org/
import requests

# API authorization
bronto_access_token = "a valid rest api access token"
headers = {"Authorization": "Bearer " + bronto_access_token}

catalog_id = "your catalog ID here"

# update multiple products
update_request_1 = {
    "fields": {
        "product_id": "product-0001",
        "parent_product_id": "None",
        "product_url": "http://mysite.com/products/0001",
        "title": "Product 0001 title",
        "description": "Product 0001 description",
        "price": "11.00 USD",
        "quantity": 47,
        "image_url": "http://mysite.com/products/0001.png",
        "additional_images": "None",
        "rating": 4.5,
        "inventory_threshold": 5,
        "availability": "Available",
        "availability_date": "",
        "product_category": "Miscellaneous",
        "sale price effective
date": "2010-01-10T00:00-0100/2020-01-10T00:00-0100",
        "tax": "US:1111:8.75, US:1111:7.75",
        "gender": "Male",
        "color": "Blue",
        "size": "XL"
    }
}

update_request_2 = {
    "fields": {
        "product_id": "product-0002",
        "parent_product_id": "None",
        "product_url": "http://mysite.com/products/0002",
        "title": "Product 0002 title",
        "description": "Product 0002 description",
```



```

        "price": "27.00 USD",
        "quantity": 25,
        "image_url": "http://mysite.com/products/0002.png",
        "additional_images": "None",
        "rating": 3.9,
        "inventory_threshold": 5,
        "availability": "Available",
        "availability_date": "",
        "product_category": "Miscellaneous",
        "sale price effective
date": "2010-01-10T00:00-0100/2020-01-10T00:00-0100",
        "tax": "US:1111:8.75, US:1111:7.75",
        "gender": "Male",
        "color": "Green",
        "size": "M"
    }
}

url_fmt = "https://rest.bronto.com/products/public/catalogs/{catalog_id}/
products"
url = url_fmt.format(catalog_id=catalog_id)

update_requests = [update_request_1, update_request_2]
response = requests.put(url, json=update_requests, headers=headers)

if response.ok:
    if response.content:
        # some errors or warnings are given
        messages = response.json()

        for message in messages:
            print(message)
    else:
        print("Products update successfully (no messages returned)")
else:
    print("Could not update products (status code {}, content='{}',
headers={})".format(
        response.status_code, response.text, response.headers
    ))

```

Java Code Example

```

import javax.ws.rs.client.Client;
import javax.ws.rs.client.ClientBuilder;
import javax.ws.rs.client.Entity;
import javax.ws.rs.core.Form;
import javax.ws.rs.core.MediaType;
import javax.ws.rs.core.Response;
import javax.ws.rs.core.UriBuilder;
import java.net.URI;
import java.util.Arrays;
import java.util.HashMap;
import java.util.List;
import java.util.Map;

public class UpdateProducts {

    // - Do not change
    -----
    private static final String BRONTO_REST_HOST = "https://
rest.bronto.com";

```

```

    private static final String RELATIVE_ENDPOINT_PATH = "products/public/
catalogs/{catalogId}/products";

    private static final String BRONTO_AUTH_PATH = "https://auth.bronto.com/
oauth2/token";

    //-----

    // Modify the below values for your specific site and catalog

    // Replace the below OAuth Request property values with your site
    specific credentials
    private static final String CLIENT_ID = "1234567890";
    private static final String CLIENT_SECRET = "0987654321";

    // Replace with your catalog ID
    private static final String CATALOG_ID = "1";

    // Simple class to represent a product update
    private static class UpdateProduct {

        private Map<String, Object> fields;

        public Map<String, Object> getFields() {
            return fields;
        }

        public void setFields(Map<String, Object> fields) {
            this.fields = fields;
        }
    }

    public static void main(String[] args) {

        Client client = ClientBuilder.newClient();

        // To be able to access Any Bronto Rest API, you need an access
token.
        // First, we build the request data needed to gain an access token
        Form requestData = new Form();
        requestData.param("grant_type", "client_credentials");
        requestData.param("client_id", CLIENT_ID);
        requestData.param("client_secret", CLIENT_SECRET);

        // Then build and send the request
        Response oauthResponse = client.target(BRONTO_AUTH_PATH)
            .request(MediaType.APPLICATION_JSON)
            .accept(MediaType.TEXT_PLAIN_TYPE)
            .post(Entity.form(requestData));

        if (oauthResponse.getStatus() != Response.Status.OK.getStatusCode())
        {
            throw new RuntimeException("Unable to get access token.");
        }

        // Retrieve the access token from the response
        Map<String, Object> responseData =
oauthResponse.readEntity(Map.class);
        String accessToken = (String) responseData.get("access_token");

        // Here begins the Update Product code
    }
}

```

```

// Create and populate some products update objects
Map<String, Object> productFields_1 = new HashMap<String, Object>();
productFields_1.put("sku", "PRO_0067");
productFields_1.put("parent", "PAR_0007");
productFields_1.put("margin", 100.0);
productFields_1.put("product_url", "http://bronto.com");
productFields_1.put("title", "Windbreaker Jacket");
productFields_1.put("description", "The Bronto windbreaker is
perfect for those windy, rainy days.");
productFields_1.put("price", "49.00 USD");
productFields_1.put("quantity", 0.0);
productFields_1.put("image_url", "http://brontogear.com/
windbreaker.jpg");
productFields_1.put("additional_images", null);
productFields_1.put("rating", 4.6);
productFields_1.put("reviews", 10);
productFields_1.put("inventory_threshold", 1);
productFields_1.put("availability", "Out of Stock");
productFields_1.put("product_category", "Miscellaneous");
productFields_1.put("Sale Price Effective Date", null);
productFields_1.put("sale_price", 39.95);
productFields_1.put("gender", "Unisex");
productFields_1.put("color", "Black");
productFields_1.put("size", "M");

Map<String, Object> productFields_2 = new HashMap<String, Object>();
productFields_2.put("sku", "PRO_0068");
productFields_2.put("parent", null);
productFields_2.put("margin", 10.0);
productFields_2.put("product_url", "http://bronto.com");
productFields_2.put("title", "Backpack");
productFields_2.put("description", "The Bronto backpack is great for
holding just about anything.");
productFields_2.put("price", "79.00 USD");
productFields_2.put("quantity", 10.0);
productFields_2.put("image_url", "http://brontogear.com/
backpack.jpg");
productFields_2.put("additional_images", null);
productFields_2.put("rating", 4.4);
productFields_2.put("reviews", 65);
productFields_2.put("inventory_threshold", 5);
productFields_2.put("availability", "In Stock");
productFields_2.put("product_category", "Miscellaneous");
productFields_2.put("Sale Price Effective Date", null);
productFields_2.put("sale_price", 69.95);
productFields_2.put("gender", "Unisex");
productFields_2.put("color", "Green");
productFields_2.put("size", null);

UpdateProduct product1 = new UpdateProduct();
UpdateProduct product2 = new UpdateProduct();
product1.setFields(productFields_2);
product2.setFields(productFields_2);

List<UpdateProduct> updateProductRequests = Arrays.asList(product1,
product2);

// Build the URL
URI uri = UriBuilder.fromUri(BRONTO_REST_HOST)
    .path(RELATIVE_ENDPOINT_PATH)
    .resolveTemplate("catalogId", CATALOG_ID)
    .build();

```

```

//Create and execute the authorized get request
Response productResponse = client.target(uri)
    .request(MediaType.APPLICATION_JSON)
    .header("Authorization", "Bearer " + accessToken)
    // Convert the request into JSON and call the PUT endpoint
    .put(Entity.json(updateProductRequests));

// Retrieve the response
String product = productResponse.readEntity(String.class);
System.out.println(product);
}
}

```

REST Campaign Service

The Campaign Service offers REST APIs for interacting with campaigns. Common calls for campaigns include Get, Create, and Update. It is accessible using HTTPS and secured with OAuth 2.0.

Create Campaigns

The campaigns POST function allows you to create a new campaign with the supplied fields.

Overview

While id, createdAt, and modifiedAt are given when retrieving a campaign object, they should be left off the POST function body because they are set by the service. The campaignTypeId, name, and archived fields are required. The description field is optional.

On success, a 201 code is returned with the created campaign.

This request requires an access token with “campaigns-read” scope.

URI

```
POST: https://rest.bronto.com/campaigns/
```

Request Body

The data used to create the campaign.

```

{
  "campaignTypeId": number,
  "name": string,
  "description": string,
  "archived": boolean
}

```

Response

A response containing the newly created campaign.

```

{
  "id": number,
  "siteId": number,
  "createdAt": ISO-8601 datetime,
  "modifiedAt": ISO-8601 datetime,
  "campaignTypeId": number,
  "name": string,
  "description": string,

```

```
"archived": boolean
}
```

Python Code Example

```
from __future__ import print_function

# Requests documentation: http://docs.python-requests.org/
import requests

# API authorization
bronto_access_token = "access token"
session = requests.Session()
session.headers["Authorization"] = "Bearer " + bronto_access_token

url = "https://rest.bronto.com/campaigns/"

json = { "campaignTypeId": 1, "name": "campaign name", "archived": False }

response = session.post(url, json=json)

if response.ok:
    campaigns = response.json()
    print("campaign:", campaigns)
else:
    print("Could not post campaign (status code={}, content='{}',
headers={})".format(
        response.status_code, response.text, response.headers
    ))
```

Get Campaign

Use a campaign ID to retrieve the details of an individual campaign.

Overview

This request requires an access token with “campaigns-read” scope.

URI

```
https://rest.bronto.com/campaigns/{campaignId}
```

Parameters

Parameter	Type	Description
campaignId	Required number	The ID for the campaign.

Response

A response containing the campaign with the given ID.

```
{
  "id": number,
  "siteId": number,
  "createdDate": org.joda.time.DateTime,
  "modifiedDate": org.joda.time.DateTime,
  "campaignTypeId": number,
  "name": string,
```

```

    "description": string,
    "archived": boolean
}

```

Python Code Example

```

from __future__ import print_function

# Requests documentation: http://docs.python-requests.org/
import requests

# API authorization
bronto_access_token = "access token"
session = requests.Session()
session.headers["Authorization"] = "Bearer " + bronto_access_token

campaign_id = "123";

url_fmt = "https://rest.bronto.com/campaigns/{campaign_id}"
url = url_fmt.format(campaign_id=campaign_id)

response = session.get(url)

if response.ok:
    campaign = response.json()
    print("campaign:", campaign)
else:
    print("Could not get campaign " + campaign_id + " (status code={},
content='{}', headers={})".format(
        response.status_code, response.text, response.headers
    ))

```

Get Campaigns

This returns a list of campaigns.

Overview

This request requires an access token with “campaigns-read” scope.

URI

```

GET: https://rest.bronto.com/campaigns?
page=1&perPage=30&sort=id&order=asc&archived=false

```

Parameters

Parameter	Type	Default Value	Description
page	Integer	1	Page to list results from.
perPage	Integer	30	Number of results displayed per page. If perPage is set to 5 and you have 10 results, then you will see 5 results on page 1 and 5 results on page 2.

Parameter	Type	Default Value	Description
sort	String	Campaign ID	Sort results by field. Possible values: <ul style="list-style-type: none"> id: campaign ID created: date the campaign was created modified: date the campaign was last modified type: campaign type (Example: Birthday) name: campaign name
order	String	Ascending	The order that results will be displayed in, either ascending or descending. Possible values: <ul style="list-style-type: none"> asc: ascending desc: descending
search	String		Search for a campaign by name. You must enter the entire campaign name.
typeIds	List		Limit results from campaigns with specific type IDs (Example query string: typeIds=1&typeIds=4).
archived	Boolean	false	Limit results to archived campaigns. Possible values: <ul style="list-style-type: none"> true false

Response

A response containing the campaign with the given ID.

```
[
  {
    "id": number,
    "siteId": number,
    "createdDate": ISO-8601 datetime,
    "modifiedDate": ISO-8601 datetime,
    "campaignTypeId": number,
    "name": string,
    "description": string,
    "archived": boolean
  }
]
```

Python Code Example

```
from __future__ import print_function

# Requests documentation: http://docs.python-requests.org/
import requests

# API authorization
```

```
bronto_access_token = "access token"
session = requests.Session()
session.headers["Authorization"] = "Bearer " + bronto_access_token

url = "https://rest.bronto.com/campaigns/"

response = session.get(url)

if response.ok:
    campaigns = response.json()
    print("campaigns:", campaigns)
else:
    print("Could not get campaigns (status code={}, content='{}',
headers={})".format(
        response.status_code, response.text, response.headers
    ))
```

Get Campaigns Types

This returns a list of all available campaign types.

Overview

This request requires an access token with “campaigns-read” scope.

URI

```
GET: https://rest.bronto.com/campaigns/type
```

Response

A response containing the campaign with the given ID.

```
[
  {
    "id":number,
    "name":string,
    "description":string
  }
]
```

Python Code Example

```
from __future__ import print_function

# Requests documentation: http://docs.python-requests.org/
import requests

# API authorization
bronto_access_token = "access token"
session = requests.Session()
session.headers["Authorization"] = "Bearer " + bronto_access_token

url = "https://rest.bronto.com/campaigns/type/"

response = session.get(url)

if response.ok:
    campaign_types = response.json()
```



```

    print("campaign types:", campaign_types)
else:
    print("Could not get campaign types (status code={}, content='{}',
          headers={})".format(
                response.status_code, response.text, response.headers
            ))

```

Update Campaigns

The campaigns PUT function allows you to update a campaign with the supplied fields.

Overview

While `createdDate`, and `modifiedDate` are given when retrieving a campaign object, they should be left off the PUT function body because they are set by the service. ID is set in the URL so you do not need to put it in the request body. The `campaignTypeId`, `name`, and `archived` fields are required. The `description` field is optional but it will replace the original `description` field with null if it is not supplied.

On success, a 201 code is returned with the created campaign.

This request requires an access token with “campaigns-read” scope.

URI

```
PUT: https://rest.bronto.com/campaigns/{campaignId}
```

Parameters

Parameter	Type	Description
<code>campaignId</code>	Required number	The ID of the campaign you want to update.

Request Body

The campaign data used to update the campaign.

```

{
  "campaignTypeId": number,
  "name": string,
  "description": string,
  "archived": boolean
}

```

Response

A response containing the updated campaign.

```

{
  "id": number,
  "siteId": number,
  "createdDate": ISO-8601 datetime,
  "modifiedDate": ISO-8601 datetime,
  "campaignTypeId": number,
  "name": string,
  "description": string,
  "archived": boolean
}

```

Python Code Example

```

from __future__ import print_function

# Requests documentation: http://docs.python-requests.org/
import requests

# API authorization
bronto_access_token = "access token"
session = requests.Session()
session.headers["Authorization"] = "Bearer " + bronto_access_token

campaign_id = "2473";

url_fmt = "https://rest.bronto.com/campaigns/{campaign_id}"
url = url_fmt.format(campaign_id=campaign_id)

json = { "campaignTypeId": 1, "name": "campaign name
2", "description": "short description", "archived": False }

response = session.put(url, json=json)

if response.ok:
    campaign = response.json()
    print("campaign:", campaign)
else:
    print("Could not put campaign (status code={}, content='{}',
headers={})".format(
        response.status_code, response.text, response.headers
    ))

```

API Errors

This is a list of the SOAP and REST API errors you may receive.

KEY	CODE	MESSAGE
General		
UNKNOWN_ERROR	101	There was an unknown API error. Please try your request again shortly.
INVALID_TOKEN	102	Authentication failed for token: %s
INVALID_SESSION_TOKEN	103	Your session is invalid. Please log in again.
INVALID_ACCESS	104	You do not have valid access for this method.
INVALID_INPUT_ARRAY	105	You must specify at least one item in the input array.
INVALID_PARAMETER	106	Unable to verify parameter %s
INVALID_REQUEST	107	There was an error in your soap request. Please examine the request and try again.
SHARD_OFFLINE	108	The API is currently undergoing maintenance. Please try your request again later.
SITE_INACTIVE	109	This site is currently marked as 'inactive'

KEY	CODE	MESSAGE
REQUIRED_FIELDS	110	Required fields are missing: %s
UNAUTHORIZED_IP	111	Your IP address does not have access for token %s.
INVALID_FILTER	112	Invalid filter type (must be AND or OR).
READ_ERROR	113	There was an error reading your query results. Please try your request again shortly.
PAGE_SIZE_EXCEEDED	114	Page size exceeds the maximum allowed limit of %d.
PAGE_SIZE_TOO_LOW	115	Page size is lower than the minimum allowed limit of %d.
END_OF_QUERY	116	End of result set.
DEPRECATED_API	117	The API function has been deprecated.
PAGE_SIZE_TOO_SMALL	118	Page size is smaller than the minimum.
SITE_LOCKED	119	This site is currently locked.
INVALID_SITE	120	The site associated with this session is invalid or does not exist.
INACTIVE_TOKEN	121	The API token you provided is inactive.
Deliveries		
INVALID_SEND_DATE	201	The send date is invalid.
INVALID_FROM_ADDRESS	202	The 'from' email address is invalid.
FATAL_ERROR_SEND	203	Fatal error sending delivery.
INVALID_RECIPIENT_TYPE	204	The recipient type is invalid: %s
INVALID_MESSAGE	205	The delivery message was not found: %s
INVALID_LIST	206	The list for this delivery was not found: %s
INVALID_SEGMENT	207	The segment for this delivery was not found: %s
INVALID_SUBSCRIBER	208	The subscriber for this delivery was not found: %s
NO_RECIPIENTS	209	Your delivery has no recipients.
INVALID_MESSAGE_RULE	210	The message rule is invalid: %s
INVALID_TRANSACTIONAL_SEND	211	Transactional sending is not enabled.
ALLOCATION_EXCEED	212	This operation would exceed your email allocation of %d
INVALID_FROM_NAME	213	The 'from' name is invalid.
INVALID_MESSAGE_FIELD	214	Invalid message field: %s
MESSAGE_NOT_TRANSACTIONAL_APPROVED	215	Message not approved for transactional sending: %s
MESSAGE_FIELD_MISSING_POSITION	216	Missing required position in message field name: %s
NONUNIQUE_MESSAGE_FIELD_POSITION	217	Position must be unique in message field name: %s

KEY	CODE	MESSAGE
MESSAGE_NOT_SENDING_APPROVED	218	Message not approved for sending: %s
NOT_FOUND	219	Delivery does not exist: %s
INVALID_THROTTLE	220	Throttle rate must be in range [0, 720] (minutes)
INVALID_REPLYTO_ADDRESS	221	The 'replyto' email address is invalid.
INVALID_STATUS	222	The delivery status was invalid: %s
INVALID_KEYWORD	223	The SMS keyword for this delivery was not found: %s
SENDING_TRIGGERED_TO_TRANSACTIONAL	224	A transactional contact cannot be sent triggered messages.
INVALID_DELIVERY_STATUS	225	Delivery status is not a valid option
INVALID_REMAIL	226	The remailObject is invalid.
REMAIL_NOT_ALLOWED	227	You cannot schedule a remail for a single send.
INVALID_DELIVERY_TYPE	228	The delivery type is invalid.
INVALID_CART_ID	229	Invalid Cart ID.
INVALID_ORDER_ID	230	Invalid Order ID.
BOTH_CART_AND_ORDER	231	Cannot assign both 'cartId' and 'orderId'.
CART_OR_ORDER_ON_BULK	232	Cart and Order IDs are only supported on single sends.
UNKNOWN_CART_ID	233	The Cart ID is not recognized.
UNKNOWN_ORDER_ID	234	The Order ID provided is not recognized.
MESSAGE_CONTENT_NOT_FOUND	235	No message content was found for the message.
MESSAGE_FIELD_TOO_LARGE	236	Message field is over 1 MB in size.
MESSAGE_FIELD_INVALID_TYPE	237	Message field given an invalid type.
DUPLICATE_BULK_DELIVERY	240	Cannot send the same message to the same recipients at the same time more than once.
TOO_MANY_PRODUCTS	241	Too many products.
PRODUCT_PLACEHOLDER_MISSING	242	Product placeholder is missing.
PRODUCT_ID_MISSING	243	Product id for placeholder X is missing.
PRODUCT_PLACEHOLDER_TOO_LONG	244	Product placeholder X is too long.
PRODUCT_ID_TOO_LONG	245	Product id for placeholder X is too long.
PRODUCT_IS_NULL	246	Product cannot be null.
Contacts		
INVALID_REQUEST	301	Invalid request: %s
NOT_FOUND	302	Contact does not exist: %s
INVALID_EMAIL	303	Invalid email address: %s
INVALID_STATUS	304	Invalid status: %s

KEY	CODE	MESSAGE
ALREADY_EXISTS	305	Contact already exists: %s
INVALID_FIELD	306	Invalid field attributes.
INVALID_MESSAGE_PREF	307	Invalid message preference: %s
ALLOCATION_EXCEED	308	This operation would exceed your contact allocation of %d
INVALID_LIST	309	The specified list is invalid: %s
INVALID_SEGMENT	310	The specified segment is invalid: %s
MAX_SEARCH_ITEMS_EXCEEDED	311	The maximum number of contact search items was exceeded (%s).
MAX_SEARCH_LISTS_EXCEEDED	312	The maximum number of contact search lists was exceeded (%s).
MAX_SEARCH_SEGMENTS_EXCEEDED	313	The maximum number of contact search segments was exceeded (%s).
EMAIL_ALREADY_EXISTS	314	Email address already exists on another contact.
EMAIL_SUPPRESSED	315	Email address is on suppression list.
FIELD_DOES_NOT_EXIST	316	The specified field does not exist (%s)
INVALID_EMAIL_LENGTH	317	Email address cannot exceed 100 characters in length: %s
MOBILE_ALREADY_EXISTS	318	Mobile number already exists on another contact.
INVALID_MOBILE	319	Invalid mobile number: %s
MISSING_EMAIL_AND_MOBILE	320	Email address or mobile number is required.
MOBILE_FAILED_TO_UPDATE	321	Subscriber was updated, mobile number update failed. (eId=%s, mobile number = %s)
MOBILE_CONTACT_ADD_PARTIAL_FAIL	322	Subscriber was added but mobile number add failed. (eId=%s, mobile number = %s)
MOBILE_ONLY_CONTACT_ADD_PARTIAL_FAIL	323	Failed to fully rollback failed insert for mobile only subscriber,” + “empty subscriber with eId=%s exists. (mobile number was %s)
SMS_KEYWORD_INVALID	324	Keyword %s is not a valid keyword. Please make sure you have added it to your account.
SMS_KEYWORD_NO_MOBILE_NUMBER	325	Cannot add a contact with sms keywords that has an empty mobile number
SMS_NOT_ENABLED_ON_SITE	326	Cannot add contacts with a keyword until SMS has been enabled for your site
CONTACT_WAS_DELETED	328	Contact was deleted and must be re-added before being updated: %s
INVALID_KEYWORD	329	The specified SMS keyword is invalid: %s
Fields		
INVALID_FIELD	401	The specified field was invalid.

KEY	CODE	MESSAGE
ALREADY_EXISTS	402	A field with this name already exists.
INVALID_DISPLAY	403	The specified field type was invalid: %s
INVALID_NAME	404	The field name was missing or invalid.
INVALID_VISIBILITY	405	The specified field visibility was invalid.
ALLOCATION_EXCEED	408	This operation would exceed your field allocation of %d.
INVALID_FIELD_VALUE	409	The value specified for the field '%s' was invalid.
DATA_TRUNCATION	410	The value specified for the field '%s' was too large.
SEGMENT_DEPENDENCY	411	Field '%s' cannot be deleted because a segment depends upon it.
DELETION_FAILURE	412	Field '%s' could not be deleted.
FIELD_USED_BY_WORKFLOW	414	Cannot delete field because it is being referenced by workflow(s): %s
Lists		
INVALID_LIST	501	The specified mail list was invalid
ALREADY_EXISTS	502	A list with this name already exists.
LIST_IS_SEGMENTED	503	This list is associated with segments.
LIST_HAS_AUTOMATORS	504	This list is associated with message rules.
LIST_HAS_DELIVERIES	505	This list is associated with deliveries.
ALREADY_ON_LIST	506	Contact is already on the list: %s
MAX_CONTACTS_EXCEEDED	507	A max of 5000 contacts may be added to a list in a single call: %s were specified
NO_CONTACTS_SPECIFIED	508	No contacts were specified.
LABEL_LENGTH_EXCEEDED	509	Label must not exceed %s characters in length.
NAME_LENGTH_EXCEEDED	510	Name must not exceed %s characters in length.
LIST_USED_BY_WORKFLOW	511	Cannot delete list because it is being referenced by workflow(s): %s
Messages		
INVALID_FOLDER_ID	601	The folder id is invalid.
INVALID_FOLDER_NAME	602	The folder name is invalid: %s
INVALID_MESSAGEGROUP	603	The specified message is invalid.
INVALID_AUTOMATOR	604	The specified automator is invalid.
INVALID_SOURCE_TEMPLATE	605	Invalid message from template: %s
INVALID_CONTENT	606	You must specify message content
INVALID_TYPE	607	Message content type must be either 'text' or 'html'.

KEY	CODE	MESSAGE
INVALID_SUBJECT	608	You must specify a message subject.
INVALID_DYNAMIC_CONTENT	609	The message's dynamic content is invalid.
INVALID_AUTOMATOR_NAME	610	The message rule name is invalid.
INVALID_AUTOMATOR_TYPE	611	The message rule type is invalid.
INVALID_AUTOMATOR_STATUS	612	The message rule status is invalid.
AUTOMATOR_EXISTS	613	A message rule with this name already exists: %s
FOLDER_EXISTS	614	A folder with this name already exists: %s
MESSAGE_EXISTS	615	A message with this name already exists: %s
INVALID_NAME	616	You must specify a message name
MESSAGE_USED_BY_WORKFLOW	617	Cannot delete message because it is being referenced by workflow(s): %s
Accounts		
INVALID_SITE	701	The account is invalid.
DUPLICATE_SITE	702	There is already an account with the name: %s
INVALID_TOKEN	703	The API token was invalid.
INVALID_TOKEN_SITE	704	The account specified for the token was invalid: %s
INVALID_TOKEN_NAME	705	The name specified for the token was invalid: %s
Delivery Groups		
INVALID_DELIVERYGROUP	801	The specified deliverygroup was invalid.
DELIVERYGROUP_NO_ID	802	No ID provided for deliverygroup.
DELIVERYGROUP_DOES_NOT_EXIST	803	Specified deliverygroup (id=%s) does not exist.
DELIVERYGROUP_ADD_FAIL	804	Failed to add deliverygroup.
DELIVERYGROUP_LIST_FAIL	805	Failed to list %s for deliverygroup (id=%s).
DELIVERYGROUP_ID_FAIL	806	Failed to find %s with id=%s in deliverygroup. NOTE: %s is replaced by the type of item you were searching for in the delivery group (message rule, delivery, or message)
DELIVERYGROUP_IDS_FAIL	807	Failed to find %s in deliverygroup. NOTE: %s is replaced by the type of item you were searching for in the delivery group (message rule, delivery, or message)
DELIVERYGROUP_DELETE_FAIL	808	Failed to remove deliverygroup.
DELIVERYGROUP_ADD_MEMBER_FAIL	809	Failed to add one or more elements to deliverygroup.
DELIVERYGROUP_DELETE_MEMBER_FAIL	810	Failed to remove element from deliverygroup.

KEY	CODE	MESSAGE
DELIVERYGROUP_SEARCH_FAIL	811	Search failed for query=%s.
DELIVERYGROUP_UPDATE_FAIL	812	Failed to update deliverygroup.
DELIVERYGROUP_CREATED_ADD_MEMBER_FAIL	813	Created deliverygroup but failed to add one or more elements to it
DELIVERYGROUP_ADD_MEMBER_FAIL_BECAUSE_NULL	814	A deliverygroup object must be provided.
Conversions		
DUPLICATE_ORDER	901	Duplicate Order Id: %s.
MISSING_AMOUNT	902	Missing required field: amount.
MISSING_QUANTITY	903	Missing required field: quantity.
INVALID_TID_HASH	904	Invalid tid hash.
INVALID_TID	905	Invalid tid. Invalid field(s) in tid: %s.
ADD_UPDATE_FAILURE	906	There was a problem calling addOrUpdateOrders. Please try again.
ID_NOT_PRESENT	907	Order ID must be present when calling addOrUpdateOrders
INVALID_SEGMENT_ID	908	The segment ID is not valid. Please try again.
INVALID_AUTOMATOR_ID	909	The automator ID is not valid. Please try again.
INVALID_LIST_ID	910	The list ID is not valid. Please try again.
INVALID_DELIVERY_ID	911	The delivery ID is not valid. Please try again.
INVALID_CONTACT_ID	912	The contact ID is not valid. Please try again.
SITE_DOES_NOT_MATCH_COOKIE	913	The cookie is not valid for this site. Please try again.
Logins		
USERNAME_IN_USE	1001	Username in use. Please choose another.
INVALID_USERNAME	1002	Username must be 5+ alphanumeric characters including underscores but cannot begin with an underscore.
SITE_DELETED	1003	Cannot add or update logins to sites marked for deletion.
USERNAME_NOT_FOUND	1004	Cannot find login by that name.
USERNAME_USED_BY_OTHER_COMPANY	1005	Cannot modify a login that is associated with a different company.
CANT_DELETE_COMPANY_USER	1006	Cannot delete a login that is attached to the company.
CANNOT_DELETE_ADMINS	1007	Cannot delete logins with admin rights.
CONTACTINFO_REQUIRED	1008	You must include contact information when adding sub accounts.

KEY	CODE	MESSAGE
CONTACTINFO_MISSING	1009	You must include firstname, lastname, email, & phone in the contact information for a new login.
INVALID_EMAIL	1010	Invalid email address set on login's contact information.
INVALID_COUNTRY	1011	Invalid country code set on login's contact information.
Workflows		
WORKFLOW_NO_ID	1101	No ID provided for workflow.
WORKFLOW_DOES_NOT_EXIST	1102	Specified workflow (id=%s) does not exist.
WORKFLOW_SEARCH_FAIL	1103	Search failed for query=%s.
WORKFLOW_ADD_FAIL	1104	Failed to add workflow.
WORKFLOW_DELETE_FAIL	1105	Failed to remove workflow
WORKFLOW_UPDATE_FAIL	1106	Failed to update workflow
WORKFLOW_LIST_FAIL	1107	Listing all workflows failed
CONTACT_ADD_FAIL	1108	Adding contacts to workflow failed.
MAX_CONTACTS_EXCEEDED	1109	A max of 5000 contacts may be added to a workflow in a single call: %s were specified
NO_CONTACTS_SPECIFIED	1110	No contacts were specified
WORKFLOW_NO_KEYWORD	1111	No keyword provided
Activities		
INVALID_START_DATE	1201	Start date is invalid: %s
INVALID_ACTIVITY_TYPE	1202	Invalid Activity types: %s
INVALID_SIZE	1203	Activity size is invalid: %s
Delivery Recipient Stat Object		
INVALID_DELIVERYID	1301	The deliveryId (%s) was invalid.
MISSING_STATS	1302	Unable to find stats for delivery target %s.
INVALID_CONTACTID	1303	The contactId (%s) was invalid.
INVALID_LISTID	1304	The listId (%s) was invalid.
INVALID_SEGMENTID	1305	The segmentId (%s) was invalid.
Header/Footer Object		
INVALID_HEADER_FOOTER	1401	The header or footer specified is invalid.
INVALID_HTML	1402	You must specify HTML content.
INVALID_TEXT	1403	You must specify TEXT content.
ALREADY_EXISTS	1404	A header or footer with this name already exists.
SMS Delivery Object		

KEY	CODE	MESSAGE
MUST_PROVIDE_KEYWORD_FOR_CONTACT	1500	Contact must be subscribed to at least one of the keyword ids provided
MUST_PROVIDE_MESSAGE_ID	1501	Message ID cannot be empty.
MUST_PROVIDE_START_DATE	1502	Start date must be specified.
MUST_PROVIDE_RECIPIENTS	1503	Must provide recipients for the SMS delivery.
INCOMPLETE_MESSAGE_FIELD	1504	A field was provided without both the messageFieldId and value specified.
FAILED_TO_SEND	1505	We were unable to add the SMS delivery. Please try again.
ALLOCATION_EXCEED	1506	This operation would exceed your SMS allocation of %d.
MUST_BE_TRANSACTIONAL	1507	Contact sends must be marked as 'transaction' for the deliveryType.
Content Tag Object		
INVALID_CONTENTTAG	1601	The content tag specified is invalid
MISSING_NAME	1602	You must specify a name
NAME_TOO_LONG	1603	Name must be 100 characters or less
INVALID_VALUE	1604	Tag value cannot contain another content tag
ALREADY_EXISTS	1605	A content tag with this name already exists
SMS Keyword Object		
KEYWORD_USED_BY_WORKFLOW	1716	Cannot delete SMS keyword because it is being referenced by workflow(s): %s
SMS Message Object		
MESSAGE_USED_BY_WORKFLOW	1806	Cannot delete SMS message because it is being referenced by workflow(s): %s