MULTIGURA

MULTIGURA Promotion API

Integration guide for partners

HOW IT WORKS

1. The participant signs up on your site or in your app.
2. Your server sends `POST /account/set`.
3. MULTIGURA creates the account, grants the license and sends an activation email.

Base URL
https://multigura.com/wp-json/promotion

Online version
multigura.com/en/api/

Date
November 2026

1. Overview

The Promotion API lets our partners give their customers, students or community members a free one-year MULTIGURA license. It is built for universities, hosting providers, agencies and communities that run a promotion or a campaign.

Registration is fully automatic: a participant signs up on your site or in your app, and MULTIGURA instantly creates their account and grants them a license. An activation email is sent to them in their language.

Every account is linked to your promotion code. When your users later buy or renew a license, you receive a share of their payments under the terms of the agreement, and a discount can be set for them as well.

To become a partner, contact us. After the agreement is signed, you receive the integration details: two requests are all the integration takes.

2. Getting started

After the agreement with MULTIGURA is signed, you receive the **id** and the **secret** of your integration. Keep the secret on your server: it must never reach a browser or a mobile app.

Base URL: **https://multigura.com/wp-json/promotion**

Every request carries a **hash**: SHA-256 of the request values and the MD5 of the secret. It proves that the request comes from you. A request with a wrong hash, or with an id that is not an active integration, gets HTTP 401.

EXAMPLE

```
// Check account (GET /account/get)
$hash = hash('sha256', $id . $email . md5($secret));

// Create account (POST /account/set)
$hash = hash('sha256', $id . $user . $email . md5($secret));
```

3. Check account: GET /account/get

Checks whether an email is already registered in MULTIGURA.

Parameter	Type	Description
`id`	int	Integration ID.
`email`	string	User email.
`hash`	string	SHA-256 of id, email and the MD5 of the secret.

If the email is free, the response has status 0 and empty data. If it is registered, the status is 1: data holds the account only when the user came through your integration, otherwise it is empty.

The company_id and company_name fields appear in data when they are filled in. The registered date is in the Asia/Tbilisi time zone (UTC+4).

EXAMPLE

```
$id     = 1;
$secret = 'YOUR_SECRET';
$email  = 'test@mail.com';

$url = 'https://multigura.com/wp-json/promotion/account/get?' . http_build_query([
    'id'    => $id,
    'email' => $email,
    'hash'  => hash('sha256', $id . $email . md5($secret))
]);

$response = json_decode(file_get_contents($url), true);

// Response
{
    "status": 1,
    "description": "The email already exists",
    "data": {
        "id": 26,
        "login": "test",
        "email": "test@mail.com",
        "registered": "2025-07-06 00:00:00",
        "active": 1,
        "promotion_code_id": 100,
        "promotion_code": "CODE"
    }
}
```

4. Create account: POST /account/set

Creates an account with a free one-year license. The body can be sent as form data or as JSON.

Parameter	Type	Description
`id`	int	Integration ID.
`user`	string	Desired login. It is converted to lowercase, and only Latin letters and digits are kept. A login that starts with a digit gets user in front, and a login that is taken gets a number at the end, so the login in the response may differ.
`email`	string	User email.
`first_name`	string	Optional. First name.
`last_name`	string	Optional. Last name.
`company_id`	string	Optional. Company identifier.
`company_name`	string	Optional. Company name.
`lang`	string	Optional. Language of the account and of its welcome email: en, ge or ru. By default en.
`hash`	string	SHA-256 of id, user, email and the MD5 of the secret. Use the user value exactly as you send it.

If the email is free, MULTIGURA creates the account and its license and, if your integration has a promotion code, attaches the code to the account. The user gets an email in the chosen language with a link to set a password. The response has status 0 and the new account in data.

If the email is already registered, nothing is created: the response is the same as in Check account.

EXAMPLE

```php
$id     = 1;
$secret = 'YOUR_SECRET';

$data = [
    'id'         => $id,
    'user'       => 'test',
    'email'      => 'test@mail.com',
    'first_name' => 'John',
    'last_name'  => 'Smith',
    'lang'       => 'en'
];

$data['hash'] = hash('sha256', $data['id'] . $data['user'] . $data['email'] . md5($secret));

$curl = curl_init('https://multigura.com/wp-json/promotion/account/set');

curl_setopt_array($curl, [
    CURLOPT_POST           => true,
    CURLOPT_POSTFIELDS     => http_build_query($data),
    CURLOPT_RETURNTRANSFER => true
]);

$response = json_decode(curl_exec($curl), true);

// Response
{
    "status": 0,
    "description": "Done",
    "data": {
        "id": 26,
        "login": "test",
        "email": "test@mail.com",
        "registered": "2025-07-06 00:00:00",
        "active": 1,
        "promotion_code_id": 100,
        "promotion_code": "CODE"
    }
}
```

5. Statuses and errors

Every response with HTTP 200 has a status:

Status	Description	Meaning
0	Done	Check account: the email is free. Create account: the account is created.
1	The email already exists	The email is registered. Nothing is created.
99	Unknown system error	The request could not be completed. Try again later, and contact us if it repeats.

A request that cannot be processed gets an HTTP error instead, in the standard WordPress format:

HTTP	Code	Reason
400	rest_missing_callback_param	A required parameter is missing.
400	rest_invalid_param	A parameter is invalid, for example the email.
401	rest_forbidden	Wrong hash, or the id is not an active integration.

EXAMPLE

```
// HTTP 400
{
    "code": "rest_invalid_param",
    "message": "Invalid parameter(s): email",
    "data": {
        "status": 400,
        "params": {
            "email": "Invalid parameter."
        }
    }
}

// HTTP 401
{
    "code": "rest_forbidden",
    "message": "Sorry, you are not allowed to do that.",
    "data": {
        "status": 401
    }
}
```