DynaMatrics

API Endpoints

POST /api/v1/challenge/create

Returns a Json response:
{
challenge: string representing the challenge table
challenge_hash: SHA-1 hash key referencing the challenge
}

POST /api/v1/challenge/answer

The content of the post must contain the SHA-1 hashed version of the answer to the challenge table using the user's DynaMatric, their username, and the hashed value of the challenge table. The username will be associated with the client by the public key passed to the server in the request signature.

username=clientname&challenge_hash=hashedChallengeTable&answer_hash=hashedChallengeAnswer

Returns a Json response:
{
answer_success: true or false
}

POST /api/v1/users.json

Create a new Dynamatric User and send them an invitation email to setup their account

```
{
  user: {
    email: 'thisisan@email.com"
  }
}
```

RESPONSE

```
{
      user:  {
                  id: '1',
                  email: 'thisisan@email.com',
                  two_factor: false,
                  confirmed: false,
                  confirmed_at: null,
                  confirmation_email_sent_at: null,
                  reset_rule_sent_at: null,
                  last_sign_in_at: null
          }
}
```

GET /api/v1/users.json?email=thisisan@email.com

Find a user by their email address, exact match

RESPONSE

```
{
      users:  [
```

```
        {
                id: '1',
                email: 'thisisan@email.com',
                two_factor: true,
                confirmed: true,
                confirmed_at: "2015-10-07 07:01:28",
                confirmation_email_sent_at: "2015-06-14 02:57:28",
                reset_rule_sent_at: "2015-06-14 02:57:28",
                last_sign_in_at: "2015-06-14 02:57:28"
          }
        ]
}
```


RESPONSE *(user not found)*


```
{
}
```


GET https://www.dynamatrics.com/api/v1/users.json

Get all Users within your user group

RESPONSE


```
{
      users:  [
        {
                id: '1',
                email: 'thisisan@email.com',
                two_factor: true,
                confirmed: true,
                confirmed_at: "2015-10-07 07:01:28",
                confirmation_email_sent_at: "2015-06-14 02:57:28",
```

```
        reset_rule_sent_at: "2015-06-14 02:57:28",
        last_sign_in_at: "2015-06-14 02:57:28"
    },
  {
        id: '2',
        email: 'another@email.com',
        two_factor: false,
        confirmed: true,
        confirmed_at: "2015-10-07 07:01:28",
        confirmation_email_sent_at: "2015-06-14 02:57:28",
        reset_rule_sent_at: false,
        last_sign_in_at: "2015-06-14 02:57:28"
    }
  ]
```

Syntax

These are the available DynaMatric rule formats. Each DynaMatric contains four rules, which are separated by a '|' character.

{1-36},{1-36},{operator}

or

{1-36},c{constant[0-9]},+

1-36 is the cell in the matrix Left to right, top to bottom. No rule can use the same cell as another rule in the set.

Available operators are:

Addition	+
Difference	-
Lesser	<
Greater	>

In a DynaMatric you have the ability to set a constant instead of a corresponding cell. Constants are prefixed by the letter c in order to not be confused with a cell. The constant must be between 0-9. Constants can only be added to a chosen cell. No other operators are valid.

This would be an example of a valid DynaMatric string:

```
"1,36,+|6,c9,+|24,c0,+|3,19,-"
```

API Security

DynaMatrics utilizes a Hash-based message authentication code (HMAC) strategy for signing all requests to our API. This is the same strategy Amazon uses to authenticate all Amazon Web Service requests (Documentation). Here's how it works:

A canonical string is first created using your HTTP headers containing the content-type, content-MD5, request URI and the timestamp. If content-type or content-MD5 are not present, then a blank string is used in their place. If the timestamp isn't present, a valid HTTP date is automatically added to the request.

1. The canonical string string is computed as follows:

   `canonical_string = 'content-type,content-MD5,request URI,timestamp'`

2. This string is then used to create the signature which is a Base64 encoded SHA1 HMAC, using the client's private secret key.
3. This signature is then added as the `Authorization` HTTP header in the form:
4. Authorization = APIAuth 'client access id':'signature from step 2'
5. On the server side, the SHA1 HMAC is computed in the same way using the request headers and the client's secret key, which is known to only the client and the server but can be looked up on the server using the client's access id that was attached in the header. The signed request expires after 15 minutes in order to avoid replay attacks.