## Introduction

SecureAuth's Authentication API embeds the SecureAuth IdP functionality into a custom application, enabling flexible workflow configurations and user interfaces. Using a RESTful API encrypted over SSL, SecureAuth IdP can validate user IDs, passwords, PINs, soft tokens, Push-to-Accept responses, and knowledge-based answers; can generate One-time Passwords (OTPs) delivered via phone call, SMS message, email message, help desk, or Push / Push-to-Accept Notification; can analyze a user's access attempt through SecureAuth's Device / Browser Fingerprinting, Adaptive Authentication, and Behavioral Biometric profile; and can evaluate IP address risk through threat intelligence data.

Each SecureAuth IdP realm can host its own uniquely configured Authentication API, enabling various workflows and registration methods.

By simply integrating an application with SecureAuth's Authentication API, enabling Multi-Factor Authentication mechanisms, and configuring Adaptive Authentication, customers can securely direct users through unique logins and interfaces without leaving the application.

### NOTE: This Authentication API Guide is specifically for SecureAuth IdP 9.0.x

Information about using Identity Management API tools can be found in the Identity Management API Guide

If using SecureAuth IdP v9.1, see Authentication API Guide and Profile Validation API Guide for information on how to configure SecureAuth Authentication API to validate end-user information and to generate OTPs for end-user authentication

See Authentication API: Send Ad hoc OTP without Existing User Profile for specific configuration steps when using ad hoc OTP delivery to users who are not registered in the directory

### Prerequisites

1. Have access to the application code
2. Have an on-premises directory with which SecureAuth IdP can integrate
3. Create a New Realm or access an existing realm in which the Authentication API will be enabled
   
   The API can be included in any realm with any Post Authentication event as long as the appropriate directory is integrated, the Registration Methods are enabled for Multi-Factor Authentication use, Adaptive Authentication settings are in place, and device fingerprinting configurations are completed (if using all features)

4. Configure the Data tab in the SecureAuth IdP Web Admin
   
   A directory integration is required for SecureAuth IdP to pull user profile information during the login process

   A directory integration is required for the Authentication API, but SecureAuth's Web Service (Multi-Data Store) option is not supported

   The Behavioral Biometrics feature only supports LDAP directory integrations, while other Authentication API features support most directory type integrations

   Ensure that the Registration Methods Profile Properties (e.g. Phone 1, Email 1, etc.) are accurately mapped to directory attributes to enable Multi-Factor Authentication workflows
NOTE: Only API section steps are required; all other Web Admin steps are optional and should be performed based on the features to be implemented.

API

API Key

1. Check **Enable API for this realm** in the **API Key** section.

2. Click **Generate Credentials** to create a new **Application ID** and **Application Key**.

   The **Application ID** and **Application Key** are unique per realm.

   In appearance the API key looks like it is comprised of **64** random characters, but it is actually comprised of **32** two-character base-16 hexadecimal values.

   This is important to note when using the API key to produce the HMAC hash.

3. Click **Select & Copy** to copy the contents from the fields.

   These values will be required in the **HTTP Header** configuration steps below.
4. Check Enable Authentication API in the API Permissions section

Click Save once the configurations have been completed and before leaving the API page to avoid losing changes

To enable Multi-Factor Authentication...

Registration Methods / Multi-Factor Methods
# Registration Configuration

## Phone Settings

<table>
<thead>
<tr>
<th>Setting</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Phone Field 1</td>
<td>Voice Only</td>
</tr>
<tr>
<td>Phone Field 2</td>
<td>Voice and SMS/Text</td>
</tr>
<tr>
<td>Phone Field 3</td>
<td>Disabled</td>
</tr>
<tr>
<td>Phone Field 4</td>
<td>Disabled</td>
</tr>
<tr>
<td>Phone/SMS Selected</td>
<td>Voice</td>
</tr>
<tr>
<td>Phone/SMS Visible</td>
<td>True</td>
</tr>
<tr>
<td>Default Phone Country Code</td>
<td></td>
</tr>
<tr>
<td>Phone Mask (RegEx)</td>
<td></td>
</tr>
</tbody>
</table>

## Email Settings

<table>
<thead>
<tr>
<th>Setting</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Email Field 1</td>
<td>Enabled (HTML)</td>
</tr>
<tr>
<td>Email Field 2</td>
<td>Enabled (HTML)</td>
</tr>
<tr>
<td>Email Field 3</td>
<td>Disabled</td>
</tr>
<tr>
<td>Email Field 4</td>
<td>Disabled</td>
</tr>
</tbody>
</table>

## Knowledge Based Settings

<table>
<thead>
<tr>
<th>Setting</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>KB Questions</td>
<td>Enabled</td>
</tr>
<tr>
<td>KB Format</td>
<td>Base 64</td>
</tr>
<tr>
<td>Number of Questions</td>
<td>2</td>
</tr>
<tr>
<td>KB Conversion</td>
<td>False</td>
</tr>
</tbody>
</table>
1. In the **Registration Configuration** section, enable at least one Multi-Factor Authentication mechanism to be utilized in the Authentication API workflow.

Refer to **Registration Methods Tab Configuration** for more information.

If not using Multi-Factor Authentication in the workflow, then no configuration is required.

The Authentication API supports the following registration methods:

- Telephony OTP
- SMS OTP
- Email OTP
- Knowledge-based Questions and Answers
- Help Desk
- OATH Token
- Push Notification and Push-to-Accept
- Static PIN

Ensure the Registration Methods Profile Properties (e.g. Phone 1, Email 1, etc.) are mapped appropriately to directory attributes in the **Data** tab to enable Multi-Factor Authentication.

Click **Save** once the configurations have been completed and before leaving the **Registration Methods / Multi-Factor Methods** page to avoid losing changes.

---

To enable Adaptive Authentication...

**Workflow**
Adaptive Authentication

Sorting Order
Drag and drop to sort the adaptive authentication factors from first (top) to last (bottom). Only enabled factors are sorted.

- IP/Country Restriction
- IP Reputation/Threat Data
- User/Group Restriction
- Geo-Velocity Restriction

Settings

<table>
<thead>
<tr>
<th>IP/Country</th>
<th>IP Reputation/Threat Data</th>
<th>User/Group</th>
<th>Geo-velocity</th>
</tr>
</thead>
</table>

Enable IP/Country Restriction

Restriction Type: IP Restriction

IP List: Allow

Failure Action: Hard stop

☐ Require user to enter username before adaptive authentication occurs
1. In the **Adaptive Authentication** section, configure the realm to enable / disable functions to be used by the application.

Refer to **Adaptive Authentication Guide** for the full configuration steps.

Be sure to select the preferred **Authentication Mode** (in the **Workflow** section), as it dictates the **suggested actions** by the API.

![Workflow Configuration](image)

Click **Save** once the configurations have been completed and before leaving the **Workflow** page to avoid losing changes.
2. In the **IP Configuration** section, set the **Public IP Address** to the appliance's public IP address or **127.0.0.1** as preferred.

   This step is required if using **Geo-Velocity** analysis to avoid errors with the Adaptive Authentication API.

   Click **Save** once the configurations have been completed and before leaving the **System Info** page to avoid losing changes.

---

To enable Phone Number Profiling...
## Phone Settings

### Phone Number Blocking

Block phone numbers from the following sources:
- [x] Cellular Telephones
- [ ] Landlines
- [ ] IP Phones
- [x] Toll-free Numbers
- [x] Premium Rate Numbers
- [ ] Pagers
- [ ] Unknown

Block phone numbers that have recently changed carriers:
- [x] Enable
  - Allow users to approve or delete a phone number that has recently changed carriers

Store carrier information in: Aux ID 2

Block or allow phone numbers by carrier or country:
- [x] Enable block/allow list
  - Define list of blocked/allowed numbers and carriers
1. In the **Phone Number Blocking** frame in the **Registration Configuration** section, enable at least one option

   - Block phone numbers from the following [selected] **sources**
   - Block phone numbers that have recently ported **carriers**
   - Block or allow phone numbers by **carrier or country**

Refer to **Phone Number Profiling Service Configuration Guide** for complete configuration steps

See **/numberprofile Endpoints** for API configuration information

Click **Save** once the configurations have been completed and before leaving the **Multi-Factor Methods** page to avoid losing changes

---

**To enable Multi-Factor Throttling...**

**Multi-Factor Methods**
Registration Configuration

Phone Settings
- Symantec VIP: [Disabled]

Multi-Factor Settings
- Inline Initialization:
  - [ ] Missing Phone
  - [ ] Missing Email
  - [ ] Missing KB Answers
  - [ ] Missing PIN
- Auto-Submit When One Avail: [Disabled]
- OTP Length: 6

Multi-Factor Throttling
- [ ] Enable multi-factor throttling
- Only allow [5] failed attempts
  in [30] Minutes for each user
- [ ] Block use of multi-factor until time limit has expired
- [ ] Lock user account after exceeding attempts
- Store attempt count in [Aux ID 1]
Multi-Factor Throttling requires SecureAuth IdP version 9.0.2 or greater

1. In the Multi-Factor Settings section, check Enable multi-factor throttling

2. Configure the settings to customize the number of allowed failed attempts and the length of the rolling time window

3. Configure the action to take upon exceeding the allowed failed attempts

4. Select a profile property to store the count of failed attempts

Refer to Multi-Factor Throttling Configuration Guide for the full configuration steps

Click Save once the configurations have been completed and before leaving the Multi-Factor Methods page to avoid losing changes
<table>
<thead>
<tr>
<th>Property</th>
<th>Source</th>
<th>Field</th>
<th>Data Format</th>
</tr>
</thead>
<tbody>
<tr>
<td>Groups</td>
<td>Default Provider</td>
<td>memberOf</td>
<td></td>
</tr>
<tr>
<td>First Name</td>
<td>Default Provider</td>
<td>givenName</td>
<td></td>
</tr>
<tr>
<td>Last Name</td>
<td>Default Provider</td>
<td>sn</td>
<td></td>
</tr>
<tr>
<td>Phone 1</td>
<td>Default Provider</td>
<td>telephoneNumber</td>
<td></td>
</tr>
<tr>
<td>Phone 2</td>
<td>Default Provider</td>
<td>mobile</td>
<td></td>
</tr>
<tr>
<td>Phone 3</td>
<td>Default Provider</td>
<td>homePhone</td>
<td></td>
</tr>
<tr>
<td>Phone 4</td>
<td>Default Provider</td>
<td>pager</td>
<td></td>
</tr>
<tr>
<td>Fingerprints</td>
<td>Default Provider</td>
<td>audio</td>
<td>Plain Binary</td>
</tr>
<tr>
<td>Push Notification Tokens</td>
<td>Default Provider</td>
<td>jpegPhoto</td>
<td>Plain Binary</td>
</tr>
<tr>
<td>OATH Tokens</td>
<td>Default Provider</td>
<td>postalAddress</td>
<td>Plain Binary</td>
</tr>
<tr>
<td>Access Histories</td>
<td>Default Provider</td>
<td>photo</td>
<td>Plain Binary</td>
</tr>
</tbody>
</table>

Add Property
1. In the **Membership Connection Settings** section, map a directory field to the **Fingerprints** property.

   In typical AD deployments the **audio** directory field is utilized.

2. Check **Writable**

   The **Fingerprints** Property can be stored as **Plain Binary** or in **JSON** format, and has distinct requirements for the LDAP directory attribute mapped to the Property based on the **Data Format** selection.

   For **Plain Binary**, these requirements must be met for the directory field that contains the fingerprint information:

   - **Length**: 4096 minimum
   - **Data Type**: Octet string (bytes)
   - **Multi-valued**

   For **JSON**, these requirements must be met for the directory field that contains the fingerprint information:

   - **Length**: No limit / undefined
   - **Data Type**: DirectoryString
   - **Multi-valued**
3. Configure the Device / Browser Fingerprinting settings to enable the use of /dfp endpoints

Refer to Device / Browser Fingerprinting - Heuristic-based Authentication for the full configuration steps
Click **Save** once the configurations have been completed and before leaving the **Data** page to avoid losing changes.

To enable Behavioral Biometrics...
1. In the Profile Fields section, map the Behavior Biometrics Property to a directory field, e.g. comment

2. Check Writable
HTTP Header

To authenticate against the API, an HTTP basic authorization header and Content-Type header are required

1. Add a Content-Type header with a value of application/json

2. Create an Authorization Header for all requests by following the steps below

Authorization Header

For GET endpoint:

1. Build a string based on the request
   
   METHOD (GET)
   DATE/TIME
   APPLICATION ID (from SecureAuth IdP Web Admin)
   PATH (API endpoint, e.g. /secureauth2/api/v1/users/<userID>/factors)

2. Create an HMAC SHA256 hash of step 1 using the Application Key (from SecureAuth IdP Web Admin)
   
   This step is executed by calling the HMAC and producing the hash value

3. Encode the HMAC SHA256 hash from step 2 in Base64

4. Concatenate the "Application ID", ":", and the "Base64 encoded HMAC SHA256 hash" from step 3
   
   ApplicationID:Base64EncodedHMACSHA256Hash

5. Encode the value from step 4 in Base64

6. Concatenate "Basic " and the "Value of Step 5"

   Basic Step5Value
GET Request Example

Step 1
GET
Wed, 08 Apr 2015 21:37:33 GMT
1b700d2e7b7b4abfa1950c656e23e81a
/secureauth2/api/v1/users/jsmith/factors

End Result: "GET
Wed, 08 Apr 2015 21:37:33 GMT
1b700d2e7b7b4abfa1950c656e23e81a
/secureauth2/api/v1/users/jsmith/factors"

Step 2
Non-printable bytes are produced in this step

Step 3
F5yqdLDJddUYOlrpBlOJBh/YCUIMVCsWejuhiCrqMmw=

Step 4
1b700d2e7b7b4abfa1950c656e23e81a:F5yqdLDJddUYOlrpBlOJBh/YCUIMVCsWejuhiCrqMmw=

Step 5
MWI3MDBkMmUtN2I3Yi00YWJmLWExOTUtMGU4NjV1MjU0NFtdOnorVGNYNG4vbFlsTmNvNjRpQkRENVJKaHF10h0UGYwaEQ4d1d4bTg2VWk9

Step 6
Basic
MWI3MDBkMmUtN2I3Yi00YWJmLWExOTUtMGU4NjV1MjU0NFtdOnorVGNYNG4vbFlsTmNvNjRpQkRENVJKaHF10h0UGYwaEQ4d1d4bTg2VWk9

End Result:
Method: GET,
Version: 1.1,
Headers: {
  Connection: Keep-Alive
  Date: Wed, 08 Apr 2015 21:37:33 GMT
  Authorization: Basic
  Host: secureauth.company.com
  Content-Length: 0
}

For POST endpoint:

1. Build a string based on the request

   METHOD (POST)
   DATE/TIME
   APPLICATION ID (from SecureAuth IdP Web Admin)
   PATH (API endpoint, e.g. /secureauth2/api/v1/auth)
   CONTENT (JSON Parameters)

2. Create an HMAC SHA256 hash of step 1 using the Application Key (from SecureAuth IdP Web Admin)

   This step is executed by calling the HMAC and producing the hash value

3. Encode the HMAC SHA256 hash from step 2 in Base64

4. Concatenate the "Application ID", "-", and the "Base64 encoded HMAC SHA256 hash" from step 3

   ApplicationID:Base64EncodedHMAC256Hash
5. Encode the value from step 4 in Base64
6. Concatenate "Basic" and the "Value of Step 5"

Basic Step5Value

**POST Request Example**

**Step 1**
POST
Wed, 08 Apr 2015 21:27:30 GMT
1b700d2e7b7b4abfa1950c865e23e81a
/secureauth2/api/v1/auth
{"user_id":"jsmith","type":"user_id"}

**End Result:**
"POST
Wed, 08 Apr 2015 21:27:30 GMT
1b700d2e7b7b4abfa1950c865e23e81a
/secureauth2/api/v1/auth
{"user_id":"jsmith","type":"user_id"}"

**Step 2**
Non-printable bytes are produced in this step

**Step 3**
D6nkepAEtk/M+cpkyWQ/hZMXZxPJ32L++5Z2a6+pB8U=

**Step 4**
1b700d2e7b7b4abfa1950c865e23e81a:D6nkepAEtk/M+cpkyWQ/hZMXZxPJ32L++5Z2a6+pB8U=

**Step 5**
MWI3MDBkMmUtN2I3Yi00YWJmLWEzOTUtMG4NjV1MjN1ODFhOkQ2bmtlcEFFdGsvTStjcGt5V1EvaFpNWp4UEozMk
wrKzVaWmE2K3BCOFU9

**Step 6**
Basic
MWI3MDBkMmUtN2I3Yi00YWJmLWEzOTUtMG4NjV1MjN1ODFhOkQ2bmtlcEFFdGsvTStjcGt5V1EvaFpNWp4UEozMk
wrKzVaWmE2K3BCOFU9

**End Result:**
Method: POST
RequestUri: 'https://secureauth.company.com/secureauth2/api/v1/auth'
Version: 1.1
Headers: {
  Connection: Keep-Alive
  Date: Wed, 08 Apr 2015 21:27:30 GMT
  Authorization: Basic
  MWI3MDBkMmUtN2I3Yi00YWJmLWEzOTUtMG4NjV1MjN1ODFhOkQ2bmtlcEFFdGsvTStjcGt5V1EvaFpNWp4UEozMk
  wrKzVaWmE2K3BCOFU9
  Expect: 100-continue
  Host: secureauth.company.com
  Content-Length: 36
  Content-Type: application/json; charset=utf-8
}
When an Authorization Header cannot be validated, one of the following responses will be returned:

<table>
<thead>
<tr>
<th>Response 1</th>
<th>Response 2</th>
</tr>
</thead>
</table>
| ```json
{
  "status": "invalid",
  "message": "Missing authentication header.",
}
``` | ```json
{
  "status": "invalid",
  "message": "Unknown authentication scheme.",
}
``` |
| ```json
{
  "status": "invalid",
  "message": "Clock skew of message is outside threshold.",
}
``` | ```json
{
  "status": "invalid",
  "message": "AppId is unknown.",
}
``` |
| ```json
{
  "status": "invalid",
  "message": "Authentication header value is empty.",
}
``` | ```json
{
  "status": "invalid",
  "message": "Authentication header has been seen before.",
}
``` |
| ```json
{
  "status": "invalid",
  "message": "Authentication header value's format should be 'appId:hash'.",
}
``` | ```json
{
  "status": "invalid",
  "message": "Invalid credentials.",
}
``` |

3. (OPTIONAL) If utilizing the Email 2-Factor Authentication method and a different language than **US English**, create an **Accept-Language** header to generate the Email OTP messages in the preferred language.

If no **Accept-Language** header is present, the Email OTP messages default to US English.
The `/users/{username}/factors` endpoint returns a list of enabled Multi-Factor Authentication methods.

By utilizing the username in the endpoint URL, SecureAuth IdP can access the user's profile and respond with the list of available Multi-Factor Authentication mechanisms.

As a `GET` endpoint, there is no body, so no JSON parameters are required.

### Definitions

- **status**: The status of user ID provided (found, not_found, invalid, etc.); will always be in response.
- **message**: Additional information regarding the status; will always be in response.
- **user_id**: The user ID provided; will always be in response, whether successful or not.
- **factors**: The list of available multi-factor authentication methods available to the user.
  - **type**: The type of method (phone, kbq, push, etc.).
  - **id**: The SecureAuth IdP Profile Property that is mapped to the directory field containing the information required to conduct the authentication (Phone1, Email2, etc.).
    - The indexed knowledge-based questions within the Knowledge-based Questions SecureAuth IdP Property (KBQ1, KBQ2, etc.).
    - A unique identifier provided to SecureAuth IdP by the mobile device during the provisioning process (for OATH and Push).
  - **value**: The information contained in the SecureAuth IdP Property / directory field (phone number, email address, device name, etc.).
  - **capabilities**: The variations available for the factor that require user selection (phone call, text message, etc.).

### Response:
### Success

```json
{
    "status": "found",
    "message": 
    "user_id": "jsmith",
    "factors": [
        {
            "type": "phone",
            "id": "Phone1",
            "value": "123-456-7890",
            "capabilities": ["call"
            ]
        },
        {
            "type": "phone",
            "id": "Phone2",
            "value": "987-654-3210",
            "capabilities": ["sms", "call"
            ]
        },
        {
            "type": "email",
            "id": "Email1",
            "value": "jsmith@company.com"
        },
        {
            "type": "kbq",
            "id": "KBQ1",
            "value": "What city were you born in?"
        },
        {
            "type": "kbq",
            "id": "KBQ2",
            "value": "What was your favorite childhood game?"
        },
        {
            "type": "kbq",
            "id": "KBQ3",
            "value": "What was your dream job as a child?"
        },
        {
            "type": "kbq",
            "id": "KBQ4",
            "value": "Who is your personal hero?"
        },
        {
            "type": "kbq",
            "id": "KBQ5",
            "value": "What is the last name of your favorite school teacher?"
        },
        {
            "type": "kbq",
            "id": "KBQ6",
            "value": "What is the name of your favorite childhood pet?"
        }
    ]
}
```

### Fail / Error

```json
{
    "status": "not_found",
    "message": "User Id was not found"
}
HTTP Status 404

{
    "status": "invalid_group",
    "message": "User Id is not associated with a valid group."
}
HTTP Status 200

{
    "status": "disabled",
    "message": "Account is disabled."
}
HTTP Status 200

{
    "status": "lock_out",
    "message": "Account is locked out."
}
HTTP Status 200

{
    "status": "password_expired",
    "message": "Password is expired."
}
HTTP Status 200

See Server Error information below
The Multi-Factor Throttling API provides protection against an attacker attempting to 1) brute force an account via trial-and-error using large numbers of OTPs and 2) disrupt service via a denial-of-service attack where an attacker attempts to overwhelm the system by generating a large number of OTPs.

The GET method retrieves the current count of Multi-Factor attempts for the given username.

The PUT method resets the count of Multi-Factor attempts to 0 (suggested after a successful authentication); the attempt count is stored in a directory attribute configured in the Web Admin (refer to Multi-Factor Throttling Configuration Guide for more information).

The thresholds for this API are configured within the Multi-Factor Methods tab of the Web Admin; any authentication attempts exceeding these thresholds are disregarded and an error message is displayed to the end-user.

GET Responses:

Success

{  "status": "found",  "message": "",  "count": <INTEGER> }

HTTP Status 200

Failure

{  "status": "not_found",  "message": "User Id was not found",  "count": "" }

HTTP Status 404

PUT Responses:

Success

{  "status": "found",  "message": "",  "count": 0 }

HTTP Status 200

Failure

{  "status": "not_found",  "message": "User Id was not found",  "count": "" }

HTTP Status 404

Endpoints for /numberprofile are used on demand to view an end-user's phone number profile and update the record if information such as carrier, network, and / or country has changed.

NOTE: For phone numbers that are roaming, SecureAuth IdP still retrieves the carrier information, upholding the carrier blocking configurations set in the Web Admin.
Definitions

**user_id**: user ID provided

**phone_number**: user phone number provided

**status**: state of the user ID or phone number provided (not_found, invalid, valid, etc.)

**message**: additional information regarding the status of the user ID or phone number; will always be in response

**providerRequestId**: unique reference identification number for the request, generated by the data provider

**portedStatus**: user phone status for the option to block phone numbers that recently changed carriers (not_ported, ported)

**originalCarrier**: the first time a POST call is made, user phone profile information from the data provider information appears here – on subsequent POST calls, user phone profile information stored in the directory appears here

**currentCarrier**: following a PUT call and a subsequent POST call, user phone profile information from the data provider appears here

**carrierCode**: 6-digit number or a concatenation of the country code and phone type

**carrier**: name of the carrier or a concatenation of the country code and phone type

**countryCode**: 2-character country code

**networkType**: phone connection source (landline, tollfree, mobile, virtual, unknown, landline_tollfree)

**carrierStatus**: 

- **status**: state of the carrier (blocked, allowed) based on the configuration defined in the Web Admin
- **reason**: explanation for the state of the carrier (network_type, ported, country, carrier) based on the configuration defined in the Web Admin

See **Phone Number Profiling Service** for configuration settings
## POST Endpoint

<table>
<thead>
<tr>
<th>JSON Parameters</th>
<th>Success Response (Phone profile is returned with this response)</th>
<th>Fail / Error Response (Phone profile is not returned with this response)</th>
</tr>
</thead>
<tbody>
<tr>
<td>{</td>
<td></td>
<td></td>
</tr>
<tr>
<td>&quot;user_id&quot;: &quot;&lt;USERNAME&gt;&quot;,</td>
<td></td>
<td></td>
</tr>
<tr>
<td>&quot;phone_number&quot;: &quot;&lt;PHONE NO.&gt;&quot; }</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Example:</td>
<td></td>
<td></td>
</tr>
<tr>
<td>{</td>
<td></td>
<td></td>
</tr>
<tr>
<td>&quot;user_id&quot;: &quot;jsmith&quot;,</td>
<td></td>
<td></td>
</tr>
<tr>
<td>&quot;phone_number&quot;: &quot;19491234567&quot; }</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Success Response

- **numberProfileResult**:
  - **providerRequestId**: "01eda1b2-d47c-4290-b1ca-6de8b2573836",
  - **internationalFormat**: "19491234567",
  - **nationalFormat**: "(949) 123-4567",
  - **countryPrefix**: "1",
  - **countryCode**: "US",
  - **countryCodeISO3**: "USA",
  - **country**: "United States of America",
  - **portedStatus**: "not_ported",
  - **validNumber**: null,
  - **reachable**: null,
  - **roamingInfo**: null,
  - **currentCarrier**: {  
    - **carrierCode**: "US-FIXED",
    - **carrier**: "United States of America Landline",
    - **countryCode**: "US",
    - **networkType**: "landline",
    - **carrierStatus**: {  
      - **status**: "blocked",
      - **reason**: null  
    }  
  }
  - **originalCarrier**: {  
    - **carrierCode**: "US-FIXED",
    - **carrier**: "T-Mobile USA, Inc.",
    - **countryCode**: "US",
    - **networkType**: "mobile",
    - **carrierStatus**: {  
      - **status**: "allowed",
      - **reason**: null  
    }  
  }
  - **ipInfo**: null,
  - **ipWarning**: null
- **status**: "valid",
- **message**: ""

### Fail / Error Response

- **status**: "not_found",
- **message**: "User Id was not found."

HTTP Status 200 (OK - Non-existing user)

- **status**: "not_found",
- **message**: "The requested resource cannot be found."

HTTP Status 404 (Not Found - Non-existing endpoint)

- **status**: "invalid",
- **message**: "Request validation failed with: PhoneNumber was not present."

HTTP Status 400 (Bad Request - Missing phone number)

- **status**: "valid",
- **message**: ""

HTTP Status 200 (OK - Invalid phone number)

- **status**: "not_found",
- **message**: "The requested resource cannot be found."

HTTP Status 404 (Not Found - Non-existing endpoint)
PUT Endpoint

<table>
<thead>
<tr>
<th>JSON Parameters</th>
<th>Success Response (Phone profile is returned with this response)</th>
<th>Fail / Error Response (Phone profile is not returned with this response)</th>
</tr>
</thead>
<tbody>
<tr>
<td>{ &quot;user_id&quot;: &quot;&lt;USERNAME&gt;&quot;, &quot;phone_number&quot;: &quot;&lt;PHONE NO.&gt;&quot;, &quot;portedStatus&quot;: &quot;not_ported&quot;, &quot;carrierInfo&quot;: { &quot;carrierCode&quot;: &quot;&lt;6-DIGIT NO. OR COUNTRY CODE + PHONE TYPE&gt;&quot;, &quot;carrier&quot;: &quot;&lt;CARRIER NAME OR COUNTRY CODE + PHONE TYPE&gt;&quot;, &quot;countryCode&quot;: &quot;&lt;2-CHARACTER COUNTRY CODE&gt;!&quot;, &quot;networkType&quot;: &quot;&lt;TYPE OF NETWORK CONNECTION&gt;&quot; } }</td>
<td>{ &quot;status&quot;: &quot;valid&quot;, &quot;message&quot;: &quot;&quot; } HTTP Status 200 (OK - Successful PUT)</td>
<td>{ &quot;status&quot;: &quot;not_found&quot;, &quot;message&quot;: &quot;User Id was not found.&quot; } HTTP Status 200 (OK - Non-existing user)</td>
</tr>
</tbody>
</table>

Example:

```
{
  "user_id": "jsmith",
  "phone_number": "19491234567",
  "portedStatus": "not_ported",
  "carrierInfo": {
    "carrierCode": "US-FIXED",
    "carrier": "T-mobile USA, Inc.",
    "countryCode": "US",
    "networkType": "mobile"
  }
}
```

/auth Endpoints

<table>
<thead>
<tr>
<th>HTTP Method</th>
<th>URI</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>POST</td>
<td>/api/v1/auth</td>
<td><a href="https://secureauth.company.com/secureauth2/api/v1/auth">https://secureauth.company.com/secureauth2/api/v1/auth</a></td>
</tr>
</tbody>
</table>

Validates auth information (e.g. username, password, etc.) and generates OTPs for authentication
Definitions

**evaluate_number**: (optional) evaluate recipient's phone number and return whether the number is valid to be used for the dispatch of an SMS or Voice OTP

Replace the `<CONTENT>` with the actual **JSON Parameter** values before sending the POST requests

- `<USERNAME>`: User ID, e.g. jsmith
- `<PASSWORD>`: User password, e.g. P@$SW0RD
- `<ANSWER>`: User answer to knowledge-based question
- `<KBQ PROPERTY>`: Indexed location of the specific KBQ being used for the authentication, e.g. KBQ2
- `<OTP>`: One-time password generated by the OATH token or on an ad hoc basis
- `<DEVICE IDENTIFIER>`: Unique identifier provided to SecureAuth IdP by the mobile device during the provisioning process (for OATH and Push)
- `<PIN NUMBER>`: User's static PIN number (e.g. 1234)
- `<PHONE PROPERTY>`: SecureAuth IdP Profile Property that is mapped to the directory field containing the required phone number, e.g. Phone1
- `<EMAIL PROPERTY>`: SecureAuth IdP Profile Property that is mapped the directory field containing the required email address, e.g. Email1
- `<UNREGISTERED PHONE>`: Phone number that is not stored in the directory
- `<UNREGISTERED EMAIL>`: Email address that is not stored in the directory
- `<HELPDESK PROPERTY>`: Help desk option being used for this authentication, e.g. HelpDesk1

In the **Registration Methods / Multi-Factor Methods** tab in the SecureAuth IdP Web Admin, there are two Help Desk authentication options to enable (HelpDesk1 and HelpDesk2)

- `<IP ADDRESS>`: IP Address of the user's device
- `<COMPANY>`: Company name, configured in the **Registration Methods / Multi-Factor Methods** tab of the Web Admin for Push-to-Accept or can be overridden in code
- `<APP>`: Application / realm name, configured in the **Registration Methods / Multi-Factor Methods** tab of the Web Admin for Push-to-Accept or can be overridden in code

### New Functions Introduced in v9.0.2

**Evaluate Phone Number for OTP**

Include the optional `evaluate_number` flag in any `/auth` request to evaluate a recipient's phone number for validity before delivering an OTP via SMS or Voice call. The criteria used to determine whether a phone number is valid or invalid for use is configured in the **Multi-Factor Methods** tab.

Refer to [Phone Number Profiling Service Configuration Guide](#) for more information
NOTE: For phone numbers that are roaming, SecureAuth IdP still retrieves the carrier information, upholding the carrier blocking configurations set in the Web Admin.

For OTP delivery, SecureAuth IdP delivers to the OTP to the carrier, and then the carrier forwards the message to the roaming partner's network.

Send Ad hoc OTP

The Send Ad hoc OTP API allows for an OTP to be dispatched on an ad hoc basis to a valid phone number or email address that is not currently stored in the directory (i.e. unregistered).

Use the **token** request parameter to define unregistered recipient phone numbers and email addresses.

Refer to **Authentication API: Send Ad hoc OTP without Existing User Profile** for specific configuration steps when using ad hoc OTP delivery to users that are not registered in the directory.

<table>
<thead>
<tr>
<th>Function</th>
<th>JSON Parameters</th>
<th>Success Response</th>
<th>Fail / Error Responses</th>
</tr>
</thead>
<tbody>
<tr>
<td>ad hoc call</td>
<td><code>{ &quot;user_id&quot;: &quot;&lt;USERNAME&gt;&quot;, &quot;type&quot;: &quot;call&quot;, &quot;token&quot;: &quot;&lt;UNREGISTERED PHONE&gt;&quot;, &quot;evaluate_number&quot;: &quot;true&quot; }</code></td>
<td>`{ &quot;status&quot;: &quot;valid&quot;, &quot;message&quot;: &quot;&quot; }</td>
<td><code>{ &quot;status&quot;: &quot;invalid&quot;, &quot;message&quot;: &quot;Request validation failed with: User Id was not present.&quot; } </code></td>
</tr>
</tbody>
</table>

**Example:**

```
{ "user_id": "jsmith", "type": "call", "token": "44396551234", "evaluate_number": "true" }
```

Note that for ad hoc OTP delivery the only supported "type" values are: sms, call, and email.

<table>
<thead>
<tr>
<th>Failed Response</th>
<th>Error Message</th>
</tr>
</thead>
<tbody>
<tr>
<td>invalid</td>
<td>Request validation failed with: User Id was not present.</td>
</tr>
<tr>
<td>not_found</td>
<td>User Id was not found.</td>
</tr>
<tr>
<td>server_error</td>
<td>Error parsing phone field.</td>
</tr>
<tr>
<td>server_error</td>
<td>The specified string is not in the form required for an e-mail address.</td>
</tr>
</tbody>
</table>

See **Server Error** information below.
### ad hoc sms
Deliver OTP via text message to a phone number not stored in the directory

```
{
    "user_id": "<USERNAME>",
    "type": "sms",
    "token": "<UNREGISTERED PHONE>"
    "evaluate_number": "true"
}
```

**Example:**
```
{
    "user_id": "jsmith",
    "type": "sms",
    "token": "44396551234"
    "evaluate_number": "true"
}
```

### ad hoc email
Deliver OTP via email to an email address not stored in the directory

```
{
    "user_id": "<USERNAME>",
    "type": "email",
    "token": "<UNREGISTERED EMAIL>"
}
```

**Example:**
```
{
    "user_id": "jsmith",
    "type": "call",
    "token": "jsmith@company.com"
}
```

### Function | JSON Parameters | Success Response | Fail / Error Response
---|---|---|---
user_id Validate user ID

```
{
    "user_id": "<USERNAME>",
    "type": "user_id"
}
```

**Example:**
```
{
    "user_id": "jsmith",
    "type": "user_id"
}
```

See Fail / Error Responses in /users endpoint table

password Validate user password

```
{
    "user_id": "<USERNAME>",
    "type": "password"
}
```

**Example:**
```
{
    "user_id": "jsmith",
    "type": "password"
}
```

```
"type": "password",
"token": "<PASSWORD>"}
}

Example:
{
  "user_id": "jsmith",
  "type": "password",
  "token": "F@SSWORD"
}

{kba
Validate knowledge-based answer

[  "user_id": "<USERNAME>",
  "type": "kba",
  "token": "<ANSWER>",
  "factor_id": "<KBQ PROPERTY>"
]

Example:
{
  "user_id": "jsmith",
  "type": "kba",
  "token": "biking",
  "factor_id": "KBQ2"
}

{oath
Validate OATH token

[  "user_id": "<USERNAME>",
  "type": "oath",
  "token": "<OTP>",
  "factor_id": "<DEVICE IDENTIFIER>"
]

Example:
{
  "user_id": "jsmith",
  "type": "oath",
  "token": "123456",
  "factor_id": "a0b1cd23e4f5g67h8ij90k"
}

{pin
Validate static PIN

[  "user_id": "<USERNAME>",
  "type": "pin",
  "token": "<PIN NUMBER>"
]

Example:
{
  "user_id": "jsmith",
  "type": "pin",
  "token": "1234"
}
### Deliver OTP via Phone Call

```json
{
    "user_id": "<USERNAME>",
    "type": "call",
    "factor_id": "<PHONE PROPERTY>",
    "evaluate_number": "true"
}
```

**Example:**

```json
{
    "user_id": "jsmith",
    "type": "call",
    "factor_id": "Phone1",
    "evaluate_number": "true"
}
```

### Deliver OTP via SMS

```json
{
    "user_id": "<USERNAME>",
    "type": "sms",
    "factor_id": "<PHONE PROPERTY>",
    "evaluate_number": "true"
}
```

**Example:**

```json
{
    "user_id": "jsmith",
    "type": "sms",
    "factor_id": "Phone1",
    "evaluate_number": "true"
}
```

### Deliver OTP via Email

```json
{
    "user_id": "<USERNAME>",
    "type": "email",
    "factor_id": "<EMAIL PROPERTY>
}
```

**Example:**

```json
{
    "user_id": "jsmith",
    "type": "email",
    "factor_id": "Email1"
}
```

- **Valid Response:**
  ```json
  {
    "status": "valid",
    "message": "",
    "user_id": "jsmith",
    "otp": "8430"
  }
  ```

- **Invalid Response:**
  ```json
  {
    "status": "invalid",
    "message": "Request validation failed with: Unknown factor id '<X>'"
  }
  ```

See Server Error information below.
### push
Deliver OTP via Push Notification

```json
{  
  "user_id": "<USERNAME>",  
  "type": "push",  
  "factor_id": "<DEVICE IDENTIFIER>"
}
```

**Example:**

```json
{  
  "user_id": "jsmith",  
  "type": "push",  
  "factor_id": "z0y9x87wv6u5t43srq2plon"
}
```

### push_accept
Deliver Push-to-Accept notification

```json
{  
  "user_id": "<USERNAME>",  
  "factor_id": "<DEVICE IDENTIFIER>",  
  "type": "push_accept",  
  "push_accept_details": {  
    "company_name": "<COMPANY>",  
    "application_description": "<APP>",  
    "enduser_ip": "<IP ADDRESS>"
  }
}
```

**Example:**

```json
{  
  "user_id": "jsmith",  
  "factor_id": "8117b6289773ed71b4XXXXX",  
  "type": "push_accept",  
  "push_accept_details": {  
    "company_name": "ACME Corp",  
    "application_description": "Salesforce",  
    "enduser_ip": "111.222.33.44"
  }
}
```

### help_desk
Deliver OTP via help desk

```json
{  
  "user_id": "<USERNAME>",  
  "type": "help_desk",  
  "factor_id": "<HELPDESK PROPERTY>"
}
```

**Example:**

```json
{  
  "user_id": "jsmith",  
  "type": "help_desk",  
  "factor_id": "HelpDesk1"
}
```
**/auth/REF_ID**

<table>
<thead>
<tr>
<th>HTTP Method</th>
<th>URI</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>GET</td>
<td>/api/v1/auth/{REF_ID}</td>
<td><a href="https://secureauth.company.com/secureauth2/api/v1/auth/f50ab2d7-178f-4421-b3ae-9f5634fa54ef">https://secureauth.company.com/secureauth2/api/v1/auth/f50ab2d7-178f-4421-b3ae-9f5634fa54ef</a></td>
</tr>
</tbody>
</table>

Checks the status of Push-to-Accept responses

When a Push-to-Accept request is made, the corresponding response contains a Reference ID, which is then appended to the /auth endpoint to continuously check whether the login request is accepted, denied, pending, or other.

**Definitions**

**status:** The status of Push-to-Accept response (found, not_found, or invalid); will always be in response

**message:** Additional information regarding the status; will always be in response

<table>
<thead>
<tr>
<th>Success Response</th>
<th>Fail / Error Response</th>
</tr>
</thead>
<tbody>
<tr>
<td>{</td>
<td></td>
</tr>
<tr>
<td>&quot;status&quot;: &quot;found&quot;,</td>
<td>{</td>
</tr>
<tr>
<td>&quot;message&quot;: &quot;ACCEPTED&quot;</td>
<td>&quot;status&quot;: &quot;not_found&quot;,</td>
</tr>
<tr>
<td>}</td>
<td>&quot;message&quot;: &quot;NOTFOUND&quot;</td>
</tr>
<tr>
<td></td>
<td>}</td>
</tr>
<tr>
<td>{</td>
<td></td>
</tr>
<tr>
<td>&quot;status&quot;: &quot;found&quot;,</td>
<td>{</td>
</tr>
<tr>
<td>&quot;message&quot;: &quot;DENIED&quot;</td>
<td>&quot;status&quot;: &quot;invalid&quot;,</td>
</tr>
<tr>
<td>}</td>
<td>&quot;message&quot;: &quot;Invalid reference ID.&quot;</td>
</tr>
<tr>
<td></td>
<td>}</td>
</tr>
<tr>
<td>{</td>
<td></td>
</tr>
<tr>
<td>&quot;status&quot;: &quot;found&quot;,</td>
<td>{</td>
</tr>
<tr>
<td>&quot;message&quot;: &quot;FAILED&quot;</td>
<td>&quot;status&quot;: &quot;&quot;</td>
</tr>
<tr>
<td>}</td>
<td>}</td>
</tr>
<tr>
<td>{</td>
<td></td>
</tr>
<tr>
<td>&quot;status&quot;: &quot;found&quot;,</td>
<td>{</td>
</tr>
<tr>
<td>&quot;message&quot;: &quot;EXPIRED&quot;</td>
<td>&quot;status&quot;: &quot;failed&quot;</td>
</tr>
<tr>
<td>}</td>
<td>}</td>
</tr>
<tr>
<td>{</td>
<td></td>
</tr>
<tr>
<td>&quot;status&quot;: &quot;found&quot;,</td>
<td>{</td>
</tr>
<tr>
<td>&quot;message&quot;: &quot;PENDING&quot;</td>
<td>&quot;status&quot;: &quot;pending&quot;</td>
</tr>
<tr>
<td>}</td>
<td>}</td>
</tr>
</tbody>
</table>

**/adaptauth Endpoint**

<table>
<thead>
<tr>
<th>HTTP Method</th>
<th>URI</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>POST</td>
<td>/api/v1/adaptauth</td>
<td><a href="https://secureauth.company.com/secureauth2/api/v1/adaptauth">https://secureauth.company.com/secureauth2/api/v1/adaptauth</a></td>
</tr>
</tbody>
</table>

Enables SecureAuth IdP Adaptive Authentication, which analyzes an end-user's profile, group, IP address, country, geo-velocity, and any risks detected by threat intelligence data

The API utilizes the information configured in the Adaptive Authentication section of the SecureAuth IdP Web Admin. Refer to Adaptive Authentication Guide for more information

SecureAuth IdP returns a response that contains the Status, Realm Workflow, and Suggested Action

The **Status** is the configured Failure Action; the **Realm Workflow** is the workflow configured in the Web Admin; and the **Suggested Action** is the suggested next step to take based on the configurations
<table>
<thead>
<tr>
<th>Status</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Continue</td>
<td>End-user continues onto the configured workflow (Failure Action: Resume auth in Web Admin)</td>
</tr>
<tr>
<td>SkipTwoFactor</td>
<td>End-user bypasses 2-Factor Authentication and moves forward to next workflow step, e.g. password (Failure Action: Step down auth in Web Admin)</td>
</tr>
<tr>
<td>TwoFactor</td>
<td>End-user undergoes additional 2-Factor Authentication (Failure Action: Step up auth in Web Admin)</td>
</tr>
<tr>
<td>Authenticated</td>
<td>End-user is taken directly to post-authentication target, bypassing additional analysis or 2-Factor Authentication (Failure Action: Post auth in Web Admin)</td>
</tr>
<tr>
<td>HardStop</td>
<td>End-user is stopped immediately in the workflow and cannot continue (Failure Action: Hard Stop in Web Admin)</td>
</tr>
<tr>
<td>Redirect</td>
<td>End-user is redirected to URL provided, e.g. another SecureAuth IdP realm (Failure Action: Redirect in Web Admin)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Suggested Action</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>2ndfactor_password</td>
<td>End-user must undergo 2-Factor Authentication and then provide password</td>
</tr>
<tr>
<td>password</td>
<td>End-user must provide password</td>
</tr>
<tr>
<td>2ndfactor</td>
<td>End-user must undergo 2-Factor Authentication</td>
</tr>
<tr>
<td>none</td>
<td>End-user is not required to perform authentication or password validation</td>
</tr>
<tr>
<td>stop</td>
<td>End-user is stopped immediately in workflow and cannot continue</td>
</tr>
<tr>
<td>redirect</td>
<td>End-user is redirected to the provided URL</td>
</tr>
</tbody>
</table>

### JSON Parameters

<table>
<thead>
<tr>
<th>Success Response</th>
<th>Failure / Error Response</th>
</tr>
</thead>
<tbody>
<tr>
<td>Continue</td>
<td></td>
</tr>
<tr>
<td>SkipTwoFactor</td>
<td></td>
</tr>
<tr>
<td>TwoFactor</td>
<td></td>
</tr>
<tr>
<td>Authenticated</td>
<td></td>
</tr>
<tr>
<td>HardStop</td>
<td></td>
</tr>
<tr>
<td>Redirect</td>
<td></td>
</tr>
</tbody>
</table>

**Example:**

```json
{
  "user_id": "jsmith",
  "parameters": {
    "ip_address": "111.222.33.44"
  }
}
```

The IP Address is not required if only performing user / group restriction; otherwise, it is required for all other functionality.
<table>
<thead>
<tr>
<th>realm_workflow</th>
<th>2ndfactor</th>
<th>suggested_action</th>
<th>2ndfactor</th>
<th>status</th>
<th>message</th>
</tr>
</thead>
<tbody>
<tr>
<td>realm_workflow</td>
<td>2ndfactor</td>
<td>suggested_action</td>
<td>2ndfactor</td>
<td>status</td>
<td>message</td>
</tr>
<tr>
<td>realm_workflow</td>
<td>2ndfactor</td>
<td>suggested_action</td>
<td>2ndfactor</td>
<td>status</td>
<td>message</td>
</tr>
<tr>
<td>realm_workflow</td>
<td>2ndfactor</td>
<td>suggested_action</td>
<td>2ndfactor</td>
<td>status</td>
<td>message</td>
</tr>
</tbody>
</table>

| realm_workflow | 2ndfactor | usernamepassword_2ndfactor | suggested_action | usernamepassword_2ndfactor | status | message |
| realm_workflow | 2ndfactor | usernamepassword_2ndfactor | suggested_action | usernamepassword_2ndfactor | status | message |
| realm_workflow | 2ndfactor | usernamepassword_2ndfactor | suggested_action | usernamepassword_2ndfactor | status | message |
| realm_workflow | 2ndfactor | usernamepassword_2ndfactor | suggested_action | usernamepassword_2ndfactor | status | message |

| realm_workflow | usernamepassword | suggested_action | password | status | message |
| realm_workflow | usernamepassword | suggested_action | password | status | message |
| realm_workflow | usernamepassword | suggested_action | password | status | message |
| realm_workflow | usernamepassword | suggested_action | password | status | message |

| realm_workflow | usernamepassword | suggested_action | password | status | message |
| realm_workflow | usernamepassword | suggested_action | password | status | message |
| realm_workflow | usernamepassword | suggested_action | password | status | message |
| realm_workflow | usernamepassword | suggested_action | password | status | message |

| realm_workflow | usernamepassword | suggested_action | password | status | message |
| realm_workflow | usernamepassword | suggested_action | password | status | message |
| realm_workflow | usernamepassword | suggested_action | password | status | message |
| realm_workflow | usernamepassword | suggested_action | password | status | message |

| realm_workflow | usernamepassword | suggested_action | password | status | message |
| realm_workflow | usernamepassword | suggested_action | password | status | message |
| realm_workflow | usernamepassword | suggested_action | password | status | message |
| realm_workflow | usernamepassword | suggested_action | password | status | message |

| realm_workflow | usernamepassword | suggested_action | password | status | message |
| realm_workflow | usernamepassword | suggested_action | password | status | message |
| realm_workflow | usernamepassword | suggested_action | password | status | message |
| realm_workflow | usernamepassword | suggested_action | password | status | message |

| realm_workflow | usernamepassword | suggested_action | password | status | message |
| realm_workflow | usernamepassword | suggested_action | password | status | message |
| realm_workflow | usernamepassword | suggested_action | password | status | message |
| realm_workflow | usernamepassword | suggested_action | password | status | message |

| realm_workflow | usernamepassword | suggested_action | password | status | message |
| realm_workflow | usernamepassword | suggested_action | password | status | message |
| realm_workflow | usernamepassword | suggested_action | password | status | message |
| realm_workflow | usernamepassword | suggested_action | password | status | message |

| realm_workflow | usernamepassword | suggested_action | password | status | message |
| realm_workflow | usernamepassword | suggested_action | password | status | message |
| realm_workflow | usernamepassword | suggested_action | password | status | message |
| realm_workflow | usernamepassword | suggested_action | password | status | message |

| realm_workflow | usernamepassword | suggested_action | password | status | message |
| realm_workflow | usernamepassword | suggested_action | password | status | message |
| realm_workflow | usernamepassword | suggested_action | password | status | message |
| realm_workflow | usernamepassword | suggested_action | password | status | message |

| realm_workflow | usernamepassword | suggested_action | password | status | message |
| realm_workflow | usernamepassword | suggested_action | password | status | message |
| realm_workflow | usernamepassword | suggested_action | password | status | message |
| realm_workflow | usernamepassword | suggested_action | password | status | message |

<p>| realm_workflow | usernamepassword | suggested_action | password | status | message |
| realm_workflow | usernamepassword | suggested_action | password | status | message |
| realm_workflow | usernamepassword | suggested_action | password | status | message |
| realm_workflow | usernamepassword | suggested_action | password | status | message |</p>
<table>
<thead>
<tr>
<th>realm_workflow</th>
<th>username</th>
<th>suggested_action</th>
<th>status</th>
<th>message</th>
</tr>
</thead>
<tbody>
<tr>
<td>realm_workflow</td>
<td>username</td>
<td>none</td>
<td>Continue</td>
<td></td>
</tr>
<tr>
<td>realm_workflow</td>
<td>username</td>
<td>2ndfactor</td>
<td>TwoFactor</td>
<td></td>
</tr>
<tr>
<td>realm_workflow</td>
<td>username</td>
<td>none</td>
<td>Authenticate</td>
<td></td>
</tr>
<tr>
<td>realm_workflow</td>
<td>username</td>
<td>stop</td>
<td>HardStop</td>
<td></td>
</tr>
<tr>
<td>realm_workflow</td>
<td>username</td>
<td>redirect</td>
<td><a href="https://example.com">https://example.com</a></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>realm_workflow</th>
<th>persistent_token</th>
<th>suggested_action</th>
<th>status</th>
<th>message</th>
</tr>
</thead>
<tbody>
<tr>
<td>realm_workflow</td>
<td>persistent_token</td>
<td>none</td>
<td>Continue</td>
<td></td>
</tr>
<tr>
<td>realm_workflow</td>
<td>persistent_token</td>
<td>2ndfactor</td>
<td>TwoFactor</td>
<td></td>
</tr>
<tr>
<td>realm_workflow</td>
<td>persistent_token</td>
<td>none</td>
<td>Authenticate</td>
<td></td>
</tr>
<tr>
<td>realm_workflow</td>
<td>persistent_token</td>
<td>stop</td>
<td>HardStop</td>
<td></td>
</tr>
<tr>
<td>realm_workflow</td>
<td>persistent_token</td>
<td>redirect</td>
<td><a href="https://example.com">https://example.com</a></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>realm_workflow</th>
<th>persistent_token</th>
<th>suggested_action</th>
<th>status</th>
<th>message</th>
</tr>
</thead>
<tbody>
<tr>
<td>realm_workflow</td>
<td>persistent_token</td>
<td>none</td>
<td>Continue</td>
<td></td>
</tr>
<tr>
<td>realm_workflow</td>
<td>persistent_token</td>
<td>2ndfactor</td>
<td>TwoFactor</td>
<td></td>
</tr>
<tr>
<td>realm_workflow</td>
<td>persistent_token</td>
<td>none</td>
<td>Authenticate</td>
<td></td>
</tr>
<tr>
<td>realm_workflow</td>
<td>persistent_token</td>
<td>stop</td>
<td>HardStop</td>
<td></td>
</tr>
<tr>
<td>realm_workflow</td>
<td>persistent_token</td>
<td>redirect</td>
<td><a href="https://example.com">https://example.com</a></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>realm_workflow</th>
<th>persistent_token</th>
<th>suggested_action</th>
<th>status</th>
<th>message</th>
</tr>
</thead>
<tbody>
<tr>
<td>realm_workflow</td>
<td>persistent_token</td>
<td>none</td>
<td>Continue</td>
<td></td>
</tr>
<tr>
<td>realm_workflow</td>
<td>persistent_token</td>
<td>2ndfactor</td>
<td>TwoFactor</td>
<td></td>
</tr>
<tr>
<td>realm_workflow</td>
<td>persistent_token</td>
<td>none</td>
<td>Authenticate</td>
<td></td>
</tr>
<tr>
<td>realm_workflow</td>
<td>persistent_token</td>
<td>stop</td>
<td>HardStop</td>
<td></td>
</tr>
<tr>
<td>realm_workflow</td>
<td>persistent_token</td>
<td>redirect</td>
<td><a href="https://example.com">https://example.com</a></td>
<td></td>
</tr>
</tbody>
</table>
/accesshistory Endpoint

<table>
<thead>
<tr>
<th>HTTP Method</th>
<th>URI</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>POST</td>
<td>/api/v1/accesshistory</td>
<td><a href="https://secureauth.company.com/secureauth2/api/v1/accesshistory">https://secureauth.company.com/secureauth2/api/v1/accesshistory</a></td>
</tr>
</tbody>
</table>

 Creates a history of end-user's logins for use in geo-velocity (Adaptive Authentication)

Once an end-user is authenticated, the information is posted to the /accesshistory endpoint, and a new entry is created and stored in the user profile. For the next login attempt, SecureAuth IdP can use the stored information to validate whether the distance traveled from the previous login to the current attempt is feasible.

If geo-velocity is not enabled for Adaptive Authentication (in the SecureAuth IdP Web Admin), then this endpoint is not necessary.

<table>
<thead>
<tr>
<th>JSON Parameters</th>
<th>Success Response</th>
<th>Failure / Error Response</th>
</tr>
</thead>
<tbody>
<tr>
<td>{</td>
<td>&quot;status&quot;: &quot;valid&quot;, &quot;message&quot;: &quot;Access History request has been processed.&quot;</td>
<td>{</td>
</tr>
<tr>
<td>&quot;user_id&quot;: &quot;&lt;USERNAME&gt;&quot;,</td>
<td></td>
<td>&quot;status&quot;: &quot;invalid&quot;, &quot;message&quot;: &quot;Access History was not saved.&quot;</td>
</tr>
<tr>
<td>&quot;ip_address&quot;: &quot;&lt;IP ADDRESS&gt;&quot;</td>
<td></td>
<td>}</td>
</tr>
</tbody>
</table>

Example:

```
{
  "user_id": "jsmith",
  "ip_address": "111.222.33.44"
}
```

/ipeval Endpoint

<table>
<thead>
<tr>
<th>HTTP Method</th>
<th>URI</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>POST</td>
<td>/api/v1/ipeval</td>
<td><a href="https://secureauth.company.com/secureauth2/api/v1/ipeval">https://secureauth.company.com/secureauth2/api/v1/ipeval</a></td>
</tr>
</tbody>
</table>

Evaluates an IP address for risk factors based on threat intelligence data

This endpoint can be used as a standalone feature rather than alongside the other Adaptive Authentication features used in the /adaptauth endpoint.

If using the /ipeval endpoint and not the /adaptauth endpoint, then no configuration is required in the Adaptive Authentication section of the SecureAuth IdP Web Admin.
### Function

**Risk IP Address Risk Evaluation**

<table>
<thead>
<tr>
<th>JSON Parameters</th>
<th>SUCCESS Response</th>
<th>FAILURE / ERROR RESPONSE</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
| {  
| "user_id": "<USERNAME>",  
| "type": "risk",  
| "ip_address": "<IP ADDRESS>"  
| }  

#### Example:

```
{
  "ip_evaluation": {
    "method": "aggregation",  
    "ip": "5.2.189.251",  
    "risk_factor": 99,  
    "risk_color": "red",  
    "risk_desc": "Extreme risk involved",  
    "geoloc": {
      "country": "Romania",  
      "country_code": "RO",  
      "region": "Iasi",  
      "region_code": "Iasi",  
      "city": "Iasi",  
      "latitude": "47.16667",  
      "longtitude": "27.6",  
      "internet_service_provider": "RCS & RDS Business",  
      "organization": "rdsnet.ro"  
    },  
    "factoring": {  
      "threatType": 99,  
      "threatCategory": 5  
    },  
    "status": "verified",  
    "message": ""  
  },  
  "status": "invalid",  
  "message": "Unknown value. Supported values are: risk."  
}  
```

This response may occur because the SecureAuth IdP appliance does not have the required license for this feature.

Contact SecureAuth Support to upgrade.

#### Risk Factor (threatType) Scores

<table>
<thead>
<tr>
<th>Threat Type (AE, IP.threatType)</th>
<th>Score</th>
<th>SecureAuth IdP Risk Category</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Anonymous Proxy</td>
<td>100</td>
<td>Extreme</td>
<td>Authentication is coming from a server that is designed to hide or anonymize the actual source IP Address</td>
</tr>
<tr>
<td>Attacker</td>
<td>99</td>
<td>Extreme</td>
<td>Indicators confirmed to host malicious content, has functioned as a command-and-control (C2) server, and / or has otherwise acted as a source of malicious activity</td>
</tr>
<tr>
<td>Compromised</td>
<td>98</td>
<td>Extreme</td>
<td>Indicators confirmed to host malicious content due to compromise or abuse – the exact time and length of compromise is unknown unless disclosed within the report</td>
</tr>
<tr>
<td>Related</td>
<td>88</td>
<td>High</td>
<td>Indicators likely related to an attack, but potentially only partially confirmed – detailed by one or more methods, like passive DNS, geo-location, and connectivity detection</td>
</tr>
<tr>
<td>Victim</td>
<td>89</td>
<td>High</td>
<td>Indicators representing an entity that has been confirmed to have been victimized by malicious activity, where actors have attempted or succeeded compromise</td>
</tr>
<tr>
<td>Uncategorized</td>
<td>80</td>
<td>High</td>
<td>Uncategorized threat</td>
</tr>
</tbody>
</table>

#### threatCategory Scores

See Server Error information below.
<table>
<thead>
<tr>
<th>Threat Category (AE.IP, threatCategory)</th>
<th>Response Value</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Anonymous Proxy</td>
<td>0</td>
<td>Authentication is coming from a server that is designed to hide or anonymize the actual source IP Address</td>
</tr>
<tr>
<td>Cyber Espionage</td>
<td>1</td>
<td>Global issue with highly sophisticated nation-states and other actors targeting military, political, and commercial interests to gain decision advantage</td>
</tr>
<tr>
<td>Hacktivism</td>
<td>2</td>
<td>Activity ranges from nuisance level to sophisticated campaigns conducted by globally coordinated actors using increasingly sophisticated tools to negatively impact revenue or damage the brand</td>
</tr>
<tr>
<td>Enterprise</td>
<td>3</td>
<td>Threats specifically targeted at Enterprise</td>
</tr>
<tr>
<td>Critical Infrastructure</td>
<td>4</td>
<td>Threats specifically targeted at Critical Infrastructure</td>
</tr>
<tr>
<td>Cyber Crime</td>
<td>5</td>
<td>Threats typically orchestrated by criminal elements for financial benefit</td>
</tr>
<tr>
<td>Vulnerability and Exploitation</td>
<td>6</td>
<td>Threats targeting known software vulnerabilities</td>
</tr>
</tbody>
</table>

**/dfp Endpoints**

The `/dfp` endpoints enable the use of SecureAuth IdP's device / browser fingerprinting, which collects unique information from an end-user's device or browser, and stores it as a "fingerprint" in the user profile. This fingerprint is then used as a comparison for future login attempts; and if a match is made, then it is used as the second factor of authentication.

To utilize these endpoints, the SecureAuth IdP realm must be configured for Device / Browser Fingerprinting. Refer to [Device / Browser Fingerprinting - Heuristic-based Authentication](#) for more information.

**/dfp/js Endpoint**
HTTP Method | URI | Example
--- | --- | ---
GET | /api/v1/dfp/js | https://secureauth.company.com/secureauth2/api/v1/dfp/js

Enables the application to retrieve the javascript reference that is required to generate fingerprints

Using the reference, end-users' devices / browsers are analyzed and most of the required information is collected; but the remaining characteristics must be provided by the application.

**Response to request:**

```javascript
{
   "src": "https://SecureAuthIdPFQDN/SecureAuthIdPRealm/assets/scripts/api/secureauth-api.js?ver=8.1.1.071"
}
```

**HTML Javascript Usage Example**

This is an example of how to use the javascript reference to capture a client-side digital fingerprint.

```html
<script>
$(function() {
    // invoke getFingerprint() method to capture
    // client-side info and serialize to JSON string.
    var serializedData = JSON.stringify(secureAuth.api.getFingerprint());

    // assign JSON string to an input for posting
    // to web server.
    $('results').val(serializedData);
});
</script>
```

**Fingerprint Components**

<table>
<thead>
<tr>
<th>Provided by Javascript Reference</th>
<th>Provided by Application</th>
</tr>
</thead>
<tbody>
<tr>
<td>fonts</td>
<td>host_address (IP Address of end-user client)</td>
</tr>
<tr>
<td>plugins</td>
<td>user_id</td>
</tr>
<tr>
<td>timezone</td>
<td>accept</td>
</tr>
<tr>
<td>video</td>
<td>accept_charset</td>
</tr>
<tr>
<td>local_storage</td>
<td>accept_encoding</td>
</tr>
<tr>
<td>session_storage</td>
<td>accept_language</td>
</tr>
<tr>
<td>ie_user_data</td>
<td></td>
</tr>
<tr>
<td>cookie_enabled</td>
<td></td>
</tr>
<tr>
<td>user_agent</td>
<td></td>
</tr>
</tbody>
</table>
## /dfp/validate Endpoint

<table>
<thead>
<tr>
<th>HTTP Method</th>
<th>URI</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>POST</td>
<td>/api/v1/dfp/validate</td>
<td><a href="https://secureauth.company.com/secureauth2/api/v1/dfp/validate">https://secureauth.company.com/secureauth2/api/v1/dfp/validate</a></td>
</tr>
</tbody>
</table>

Compares the presented fingerprint with those stored in the user profile

Based on the information provided from the directory, SecureAuth IdP returns a response stating whether the fingerprint is found or not found. If the fingerprint is found, then the end-user has completed 2-Factor Authentication, and no additional authentication steps are required for identity validation.

### Definitions

- **fingerprint_id**: GUID of the fingerprint
- **fingerprint_name**: Descriptive name derived from the user_agent string
- **score**: Match score of the provided fingerprint (out of 100.00)
- **match_score**: Minimum score allowed to be accepted as second factor authentication (configured in realm)
- **update_score**: Minimum score allowed to be accepted to update existing fingerprint with new information (configured in realm, requires 2-Factor Authentication for update)
- **status**: Status of the matching outcomes:
  - **found**: Fingerprint found with acceptable match_score
  - **found_for_update**: Fingerprint found with unacceptable match_score, but acceptable update_score and can be updated by posting to `/dfp/confirm` endpoint
  - **found_with_id_mismatch**: Fingerprint found, but with different Fingerprint ID
  - **not_found**: Fingerprint not found and new one must be created by posting to `/dfp/confirm` endpoint
- **message**: Additional information regarding the status

### Function | JSON Parameters | Response

- **Function**: The function is not specified in the provided content.
- **JSON Parameters**: The JSON parameters are not specified in the provided content.
- **Response**: The response information is not specified in the provided content.
Validate NEW Fingerprint

```
{
  "user_id": "<USERNAME>",
  "host_address": "<IP ADDRESS>",
  "fingerprint": {
    "fonts": "<FONT LIST>",
    "plugins": "<PLUGIN LIST>",
    "timezone": "<TIMEZONE>",
    "video": "<VIDEO>",
    "local_storage": "<T OR F>",
    "session_storage": "<T OR F>",
    "ie_user_data": "<T OR F>",
    "cookie_enabled": "<T OR F>",
    "user_agent": "<USER AGENT INFO>",
    "accept": "<ACCEPT INFO>",
    "accept_charset": "<CHARSET INFO>",
    "accept_encoding": "<ENCODING INFO>",
    "accept_language": "<LANG INFO>
  }
}
```

Example:

```
{
  "user_id": "jsmith",
  "host_address": "111.222.33.44",
  "fingerprint": {
    "fonts": "Agency FB,Aharoni,Algerian,Andalus,Angsana
New,AngsanaUPC ...",
    "plugins": "ActiveTouch General Plugin Container:ActiveTouch General Plugin Container Version 105 ...",
    "timezone": "America/Los_Angeles",
    "video": "1920x1080x24",
    "local_storage": "false",
    "session_storage": "false",
    "ie_user_data": "false",
    "cookie_enabled": "true",
    "user_agent": "Mozilla/5.0 (Windows NT 6.1; WOW64; rv:41.0) Gecko/20100101 Firefox/41.0",
    "accept": "text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8",
    "accept_charset": "",
    "accept_encoding": "gzip, deflate",
    "accept_language": "en-US,en;q=0.5"
  }
}
```

Refer to the table in the /dfp/js endpoint section for more information on the required fingerprint components

Validate KNOWN Fingerprint

```
{
  "fingerprint_id": "5860f98642fe4354ba65a35a0d8bc6ef",
  "fingerprint_name": "Windows 7 - Firefox 41.0",
  "score": "100.00",
  "match_score": "95.00",
  "update_score": "80.00",
  "status": "found",
  "message": ""
}
```

```
{
  "fingerprint_id": "5860f98642fe4354ba65a35a0d8bc6ef",
  "fingerprint_name": "Windows 7 - Firefox 41.0",
  "score": "100.00",
  "match_score": "95.00",
  "update_score": "80.00",
  "status": "found",
  "message": ""
}
```
Example:

```json
{
  "user_id": "jsmith",
  "host_address": "111.222.33.44",
  "fingerprint_id": "58f0f98642fe4354ba65a35a0d8bc6ff",
  "fingerprint": {
    "fonts": "Agency FB,Aharoni,Algerian,Andalus,Angsana New,AngsanaUPC ...",
    "plugins": "ActiveTouch General Plugin Container: ActiveTouch General Plugin Container Version 105 ...",
    "timezone": "America/Los_Angeles",
    "video": "1920x1080x24",
    "local_storage": "false",
    "session_storage": "false",
    "ie_user_data": "false",
    "cookie_enabled": "true",
    "user_agent": "Mozilla/5.0 (Windows NT 6.1; WOW64; rv: 41.0) Gecko/20100101 Firefox/41.0",
    "accept": "text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8",
    "accept_charset": "",
    "accept_encoding": "gzip, deflate",
    "accept_language": "en-US,en;q=0.5"
  }
}
```

### /dfp/confirm Endpoint

<table>
<thead>
<tr>
<th>HTTP Method</th>
<th>URI</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>POST</td>
<td>/api/v1/dfp/confirm</td>
<td><a href="https://secureauth.company.com/secureauth2/api/v1/dfp/confirm">https://secureauth.company.com/secureauth2/api/v1/dfp/confirm</a></td>
</tr>
</tbody>
</table>

Stores the new or updated fingerprints in the user profile in the directory

If a fingerprint posted to the /dfp/validate endpoint returns a not_found or found_for_update status, then the information must be posted to the /dfp/confirm endpoint to create / update the fingerprint

Once the fingerprint is validated, SecureAuth IdP returns a fingerprint_id, which is then posted to the confirm endpoint to create the entry

<table>
<thead>
<tr>
<th>JSON Parameters</th>
<th>Success Response</th>
<th>Failure / Error Response</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Example:

```json
{
  "user_id": "jsmith",
  "fingerprint_id": "58f0f98642fe4354ba65a35a0d8bc6ff",
  "fingerprint": "Windows 7 - Firefox 41.0",
  "score": "90.00",
  "match_score": "95.00",
  "update_score": "80.00",
  "status": "found_for_update",
  "message": ""
}
```

```json
{
  "fingerprint_id": "a31332450f284e9bbb1572e7c1c4927a",
  "fingerprint_name": "Windows 7 - Firefox 41.0",
  "score": "100.00",
  "match_score": "95.00",
  "update_score": "80.00",
  "status": "found_with_id_mismatch",
  "message": ""
}
```

```json
{
  "fingerprint_id": "58f0f98642fe4354ba65a35a0d8bc6ff",
  "fingerprint_name": "Windows Vista - Firefox 41.0",
  "status": "found",
  "message": "Fingerprint exists.",
  "user_id": "jsmith"
}
```

```json
{
  "fingerprint_id": "58f0f98642fe4354ba65a35a0d8bc6ff",
  "status": "not_found",
  "message": "Could not resolve fingerprint with ID '38f0eef7e01004f18f7f793c7'.",
  "user_id": "jsmith"
}
```
If a server error is encountered, then the follow response is returned:

```json
{
  "status": "server_error",
  "message": "<Exception message describing the issue.>",
}
```

HTTP Status 500

/behaviorbio Endpoints

Refer to Behavioral Biometrics Guide for more information on the feature

/behaviorbio/js Endpoint

<table>
<thead>
<tr>
<th>HTTP Method</th>
<th>URI</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>GET</td>
<td>/api/v1/behaviorbio/js</td>
<td><a href="https://secureauth.company.com/secureauth2/api/v1/behaviorbio/js">https://secureauth.company.com/secureauth2/api/v1/behaviorbio/js</a></td>
</tr>
</tbody>
</table>

Enables the application to retrieve the javascript reference that is required to gather and analyze a user's behavioral biometric profile.

Using the reference, most of the data required for the POST and PUT endpoints are provided; but the remaining information (user ID, host address, and user agent) must be provided by the application.

Response to request:

```json
{
  "src": "https://SecureAuthIdPFQDN/SecureAuthIdPRealm/assets/scripts/api/behaveBio.obf.js?ver=9.0.0.22"
}
```
### Code Example

```html
<form>
  ...
  <input id="password" class="form-control" type="password" name="loginForm:Password" autocomplete="off" value="" placeholder="Password">
  <input type="hidden" name="behavio_hidden" id="behavio_hidden">
  <input id="textbox" class="form-control" type="text" name="loginForm:NotMonitored" autocomplete="off" value="" placeholder="Ignored">
  ...
</form>

<script>
$(document).ready(function() {
  secureAuth.api.behaveBio(
    anonymousByName: [],
    anonymousById: [],
    anonymousByType: ['password'],
    behavioHiddenId: ['behavio_hidden'],
    ignoreFields: ['loginForm:NotMonitored'],
    haveMouse: false
  );
});
</script>
```

### Code Explanation

#### Form

- `<input type="hidden" name="behavio_hidden" id="behavio_hidden">` – Required anywhere in the `<form>` to create the behaviorProfile, with any preferred `name` and `id` values
- `<input id="password" class="form-control" type="password" name="loginForm:Password" autocomplete="off" value="" placeholder="Password">` – Example of input field with necessary components `id`, `type`, and `name` to identify anonymous input fields
- `<input id="textbox" class="form-control" type="text" name="loginForm:NotMonitored" autocomplete="off" value="" placeholder="Not for Profile">` – Example of input field with necessary components `id`, `type`, and `name` to identify ignored fields

#### Script

Full script required in app code

- `anonymousByName` – Identifies anonymous input fields by `name` (in `<input>`) 
- `anonymousById` – Identifies anonymous input fields by `id` (in `<input>`) 
- `anonymousByType` – Identifies anonymous input fields by `type` (in `<input>`, as shown in code example) 
- `behaioHiddenId` – Calls to `<input type="hidden" ...>` line (required in `<form>`) with the same `id` value 
- `ignoreFields` – Identifies fields to not include in the behaviorProfile by `name` (in `<input>`, as shown in code example) 
- `haveMouse` – Monitors mouse movements as well as keystroke behavior (`true` or `false`)

### /behavebio Endpoint

<table>
<thead>
<tr>
<th>HTTP Method</th>
<th>URI</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>POST</td>
<td>/api/v1/behavebio</td>
<td><a href="https://secureauth.company.com/secureauth2/api/v1/behavebio">https://secureauth.company.com/secureauth2/api/v1/behavebio</a></td>
</tr>
<tr>
<td>PUT</td>
<td>/api/v1/behavebio</td>
<td><a href="https://secureauth.company.com/secureauth2/api/v1/behavebio">https://secureauth.company.com/secureauth2/api/v1/behavebio</a></td>
</tr>
</tbody>
</table>

The `POST` method collects and creates the user's behavioral biometric profile
The **PUT** method resets the user's profile to enable retraining.

**POST Endpoint**

At this endpoint, the user's behavioral biometric profile is created and then analyzed against subsequent profile information posted to the endpoint.

To utilize Behavioral Biometrics as a second factor, training must occur, which is simply the API gathering user keystroke, text input, and other factors while they fill out the application's provided fields (e.g. username, password, home address, phone number, etc.). Once a profile has been trained, the API can then return a score based on the comparison of the "normal" behavior and the current behavior. **To achieve a fully trained profile, the fields must be filled out ten (10) times.**

**Definitions**

- **userID**: User's username
- **behaviorProfile**: User's behavioral biometric profile, generated by the javascript reference acquired in the `/behavebio/js` endpoint
- **hostAddress**: User's IP Address
- **userAgent**: User's user agent (computer / browser information)
- **TotalScore**: How similar the presented behavior is to the stored profile
- **TotalConfidence**: How confident the API is that the score is accurate
- **Device**: Type of device user employs
- **Results**: Breakdown of specific field scores
### JSON Parameters

- `userId`: `<USERNAME>`
- `behaviorProfile`: `<BEHAVIORAL BIOMETRIC PROFILE>`
- `hostAddress`: `<IP ADDRESS>`
- `userAgent`: `<USER AGENT>`

### Example:

```json
{
  "userId": "jsmith",
  "behaviorProfile": 
    "["m","n",null,"mozPay":null,"mozContacts":{},"mozApps":{},"doNotTrack": " unspecified","battery":{},"oscpu": "Intel+Mac OSX+10.10","vendor": ",")
  "productSub": "20100101",
  "cookieEnabled": true,
  "buildID": "20160210153822",
  "mediaDevices": {}
}
```

### Success Response

**Untrained Profile:**

```json
{
  "BehaviorBioResults": {
    "TotalScore": 0.5,
    "TotalConfidence": 0,
    "Device": "DESKTOP",
    "Results": []
  },
  "status": "valid",
  "message": ""
}
```

**Trained Profile:**

```json
{
  "BehaviorBioResults": {
    "TotalScore": 0.999999780,
    "TotalConfidence": 0.999780,
    "Device": "DESKTOP",
    "Results": [
      {
        "ControlID": "text#UserID",
        "Score": 0.99997949,
        "Confidence": 0.9997949,
        "Count": 10
      },
      {
        "ControlID": "password#Password",
        "Score": 0.99997807,
        "Confidence": 0.9997807,
        "Count": 10
      }
    ],
    "status": "valid",
    "message": ""
}
```

### Failure / Error Response

- **Untrained Profile:**

  ```json
  {
    "status": "invalid",
    "message": "Request validation failed with: userId was not present."
  }
  ```

- **Trained Profile:**

  ```json
  {
    "status": "invalid",
    "message": "Request validation failed with: behaviorProfile was not present."
  }
  ```

- **Untrained Profile:**

  ```json
  {
    "status": "invalid",
    "message": "Request validation failed with: hostAddress was not present."
  }
  ```

- **Trained Profile:**

  ```json
  {
    "status": "invalid",
    "message": "Request validation failed with: userAgent was not present."
  }
  ```
**PUT Endpoint**

At this endpoint, the user's trained profile information can be reset; this is especially useful after a user changes a password.

A profile can be completely reset, or specific fields from which behavior profile information is collected can be reset individually.

**Definitions**

- **fieldName**: Name of field to reset (unique to application); or set to ALL for global reset.
- **fieldType**: Type of field, either **regulartext** (actual values stored in profile) or **anonymoustext** (no actual values stored in profile, e.g. password entries); or set to ALL for global reset.
- **deviceType**: Type of device used by user (**Desktop** or **Mobile**); or set to ALL for global reset.

<table>
<thead>
<tr>
<th>JSON Parameters</th>
<th>Success Response</th>
<th>Failure / Error Response</th>
</tr>
</thead>
<tbody>
<tr>
<td>{</td>
<td>{</td>
<td>{</td>
</tr>
<tr>
<td>&quot;userId&quot;: &quot;&lt;USERNAME&gt;“,</td>
<td>&quot;status&quot;: &quot;valid&quot;,</td>
<td>&quot;status&quot;: &quot;invalid&quot;,</td>
</tr>
<tr>
<td>&quot;fieldName&quot;: &quot;&lt;FIELD TO RESET&gt;“,</td>
<td>&quot;message&quot;: &quot;Reset sent to data store&quot;</td>
<td>&quot;message&quot;: &quot;Request validation failed with: userId was not present.&quot;</td>
</tr>
<tr>
<td>&quot;fieldType&quot;: &quot;&lt;TYPE OF FIELD&gt;“,</td>
<td></td>
<td>}</td>
</tr>
<tr>
<td>&quot;deviceType&quot;: &quot;&lt;TYPE OF DEVICE&gt;“</td>
<td>}</td>
<td>}</td>
</tr>
</tbody>
</table>

**Specific Field Reset Example:**

```json
{
    userId: "jsmith",
    fieldName: "UserID",
    fieldType: "regulartext",
    deviceType: "DESKTOP"
}
```

**Global Reset Example:**

```json
{
    userId: "jsmith",
    fieldName: "ALL",
    fieldType: "ALL",
    deviceType: "ALL"
}
```