Authentication API 8.1 Configuration Guide

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, soft tokens, and knowledge-based answers; can generate One-time Passwords (OTPs) delivered via phone call, SMS message, email message, help desk, or PUSH Notification; 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 and enabling 2-Factor Authentication mechanisms, customers can securely direct users through unique logins and interfaces without leaving the application.

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 and the registration methods are enabled for 2-Factor Authentication use

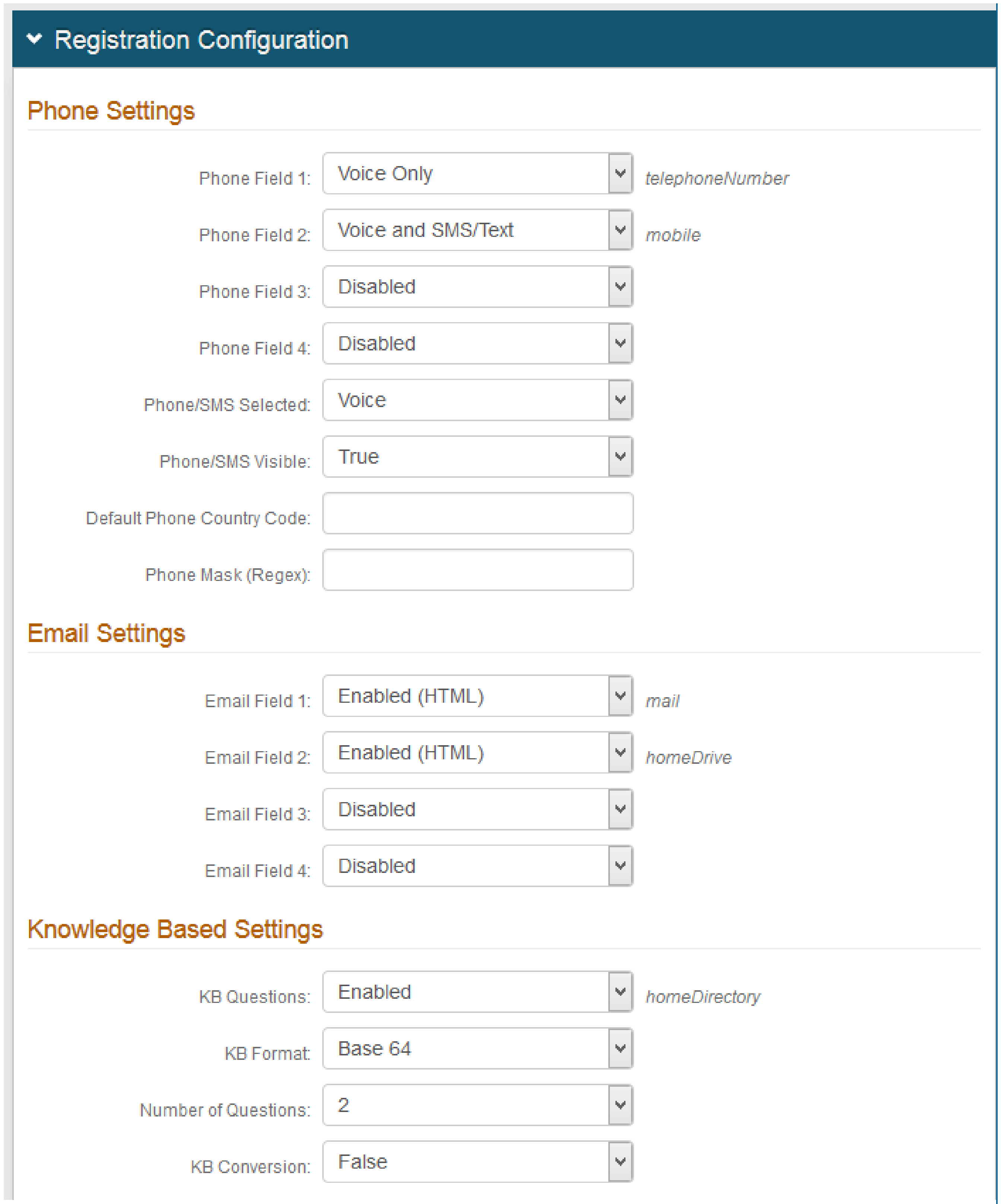
4. Configure the **Data** tab in the SecureAuth IdP Web Admin

   A directory integration is required for SecureAuth IdP to pull user profile information during the login process

   Ensure that the Registration Methods Profile Properties (e.g. Phone 1, Email 1, etc.) are accurately mapped to directory attributes to enable 2-Factor Authentication workflows

SecureAuth IdP Web Admin

Registration Methods



Registration Configuration

Phone Settings

Phone Field 1:	Voice Only	*telephoneNumber*
Phone Field 2:	Voice and SMS/Text	*mobile*
Phone Field 3:	Disabled	
Phone Field 4:	Disabled	
Phone/SMS Selected:	Voice	
Phone/SMS Visible:	True	
Default Phone Country Code:		
Phone Mask (Regex):		

Email Settings

Email Field 1:	Enabled (HTML)	*mail*
Email Field 2:	Enabled (HTML)	*homeDrive*
Email Field 3:	Disabled	
Email Field 4:	Disabled	

Knowledge Based Settings

KB Questions:	Enabled	*homeDirectory*
KB Format:	Base 64	
Number of Questions:	2	
KB Conversion:	False	

1. In the **Registration Configuration** section, enable at least one 2-Factor Authentication mechanism to be utilized in the Authentication API workflow

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

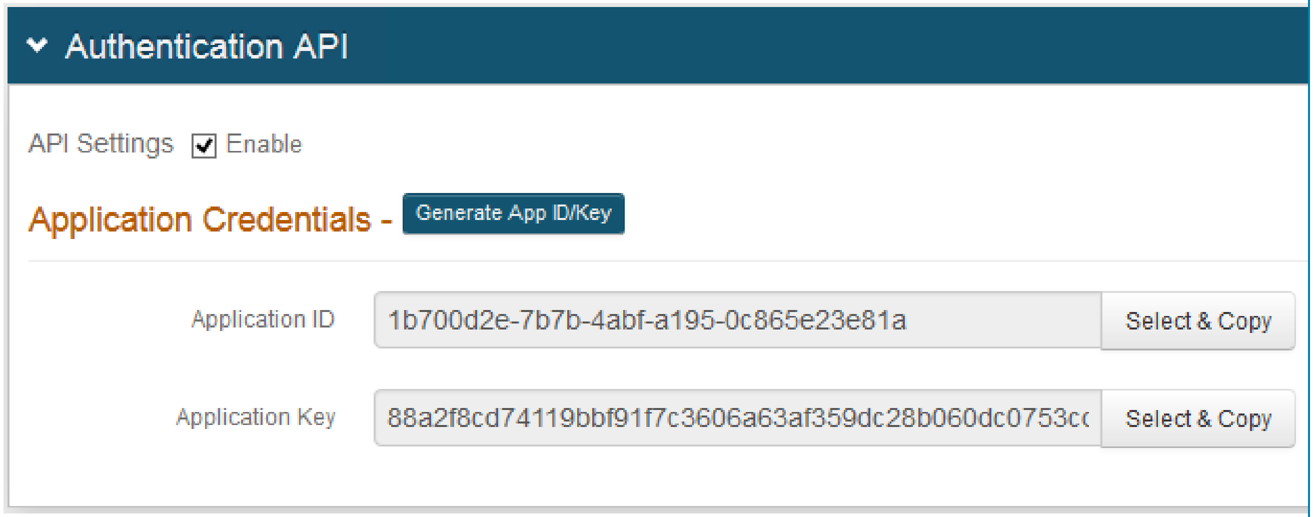
This is an optional configuration step and is only required if the workflow requires 2-Factor Authentication

If no 2-Factor Authentication is utilized in the workflow, skip ahead to step 2

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

Authentication API


Authentication API

API Settings ☑ Enable

Application Credentials - Generate App ID/Key

Application ID: 1b700d2e-7b7b-4abf-a195-0c865e23e81a

Application Key: 88a2f8cd74119bbf91f7c3606a63af359dc28b060dc0753cc

2. Check **Enable** in the **API Settings** section

3. Click **Generate App ID / Key** to create a new **Application ID** and **Application Key**

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

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

These values will be required in the **HTTP Header** configuration steps below

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

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)

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
    1b700d2e7b7b4abfa1950c865e23e81a
    /secureauth2/api/v1/users/jbeam/factors

End Result: "GET\nWed, 08 Apr 2015 21:37:33 GMT\n1b700d2e7b7b4abfa1950c865e23e81a\n
/secureauth2/api/v1/users/jbeam/factors"

Step 3
    F5yqdLDJddUYOlrpBlOJBh/YCUIMVCsWejuhiCrqMmw=

Step 4
    1b700d2e7b7b4abfa1950c865e23e81a:F5yqdLDJddUYOlrpBlOJBh/YCUIMVCsWejuhiCrqMmw=

Step 5

MWI3MDBkMmUtN2I3Yi00YWJmLWExOTUtMGM4NjVlMjNlODFhOnorVGNYNG4vbFlsTmNvNjRpQkRENVJKaHFiZ0h0UG
YwaEQ4d1d4bTgvWVk9

Step 6
    Basic
MWI3MDBkMmUtN2I3Yi00YWJmLWExOTUtMGM4NjVlMjNlODFhOnorVGNYNG4vbFlsTmNvNjRpQkRENVJKaHFiZ0h0UG
YwaEQ4d1d4bTgvWVk9

End Result:
    Method: GET,
    RequestUri: 'https://secureauth.company.com/secureauth2/api/v1/users/jbeam/factors',
    Version: 1.1,
    Headers: {
        Connection: Keep-Alive
        Date: Wed, 08 Apr 2015 21:37:33 GMT
        Authorization: Basic
MWI3MDBkMmUtN2I3Yi00YWJmLWExOTUtMGM4NjVlMjNlODFhOkY1eXFkTERKZGRVWU9scnBCbE9KQmgvWUNVSU1WQ3
NXZWp1aGlDcnFNbXc9
        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)

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":"jbeam","type":"user_id"}

End Result: "POST\nWed, 08 Apr 2015 21:27:30 GMT\n1b700d2e7b7b4abfa1950c865e23e81a\n
/secureauth2/api/v1/auth\n{"user_id":"jbeam","type":"user_id"}"

Step 3
    D6nkepAEtk/M+cpkyWQ/hZMXZxPJ32L++5ZZa6+pB8U=

Step 4
    1b700d2e7b7b4abfa1950c865e23e81a:D6nkepAEtk/M+cpkyWQ/hZMXZxPJ32L++5ZZa6+pB8U=

Step 5

MWI3MDBkMmUtN2I3Yi00YWJmLWExOTUtMGM4NjVlMjNlODFhOkQ2bmtlcEFFdGsvTStjcGt5V1EvaFpNWFp4UEozMk
wrKzVaWmE2K3BCOFU9

Step 6
    Basic
MWI3MDBkMmUtN2I3Yi00YWJmLWExOTUtMGM4NjVlMjNlODFhOkQ2bmtlcEFFdGsvTStjcGt5V1EvaFpNWFp4UEozMk
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
MWI3MDBkMmUtN2I3Yi00YWJmLWExOTUtMGM4NjVlMjNlODFhOkQ2bmtlcEFFdGsvTStjcGt5V1EvaFpNWFp4UEozMk
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:

`{ "status": "invalid", "message": "Missing authentication header.", }`	`{ "status": "invalid", "message": "Unknown authentication scheme.", }`
`{ "status": "invalid", "message": "Clock skew of message is outside threshold.", }`	`{ "status": "invalid", "message": "AppId is unknown.", }`
`{ "status": "invalid", "message": "Authentication header value is empty.", }`	`{ "status": "invalid", "message": "Authentication header has been seen before.", }`
`{ "status": "invalid", "message": "Authentication header value's format should be 'appId:hash'.", }`	`{ "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

GET Endpoint

Endpoint: **https:// SecureAuthIdPFQDN/SecureAuthIdPRealm/api/v1/users/<username>/factors**

For example: **https://secureauth.company.com/secureauth2/api/v1/users/jbeam/factors**

The **users GET** endpoint provides to the end-user the list of enabled 2-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 2-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

```
{
  "status": "found",
  "message": "",
  "user_id": "jbeam",
  "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": "jbeam@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?"
    },
    {
      "type": "help_desk",
      "id": "HelpDesk1",
      "value": "987-654-3210"
    },
    {
      "type": "help_desk",
      "id": "HelpDesk2",
      "value": "987-654-3211"
    },
    {
      "type": "push",
      "id": "b8d153fc28044a9abd87fc3657c0f443",
      "value": "HTC One"
    }
    {
      "type": "oath",
      "id": "20042165adb742d9ac1963dab156b49e",
      "value": "HTC One"
    }
  ]
}
```

Fail / Error

```
{
  "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": "invalid",
  "message": "User Id was not found."
}
HTTP Status 404
```

```
{
  "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

POST Endpoints

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

<USERNAME>: User ID, e.g. jbeam
<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
<DEVICE IDENTIFIER>: Unique identifier provided to SecureAuth IdP by the mobile device during the provisioning process (for OATH and PUSH)
<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
<HELPDESK PROPERTY>: Help desk option being used for this authentication, e.g. HelpDesk1
In the **Registration 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

Auth POST Endpoint

Endpoint: **https://SecureAuthIdPFQDN/SecureAuthIdPRealm/api/v1/auth**

For example: **https://secureauth.company.com/secureauth2/api/v1/auth**

The **auth POST** endpoint enables SecureAuth IdP to validate information, such as username and password, and generate OTPs for authentication

Function	JSON Parameters	Success Response	Fail / Error Response

user_id
 	
Validate user ID
	`{ "user_id": "<USERNAME>", "type": "user_id" }`
	
	Example:
	
	`{ "user_id": "jbeam", "type": "user_id" }`
		`{ "status": "found", "message": "User Id found", }`	See **Fail / Error Responses** in **users GET** endpoint table

password
 	
Validate user password
	`{ "user_id": "<USERNAME>", "type": "password", "token": "<PASSWORD>" }`
	
	Example:
	
	`{ "user_id": "jbeam", "type": "password", "token": "P@$SW0RD" }`
		`{ "status": "valid", "message": "" }`	`{ "status": "invalid", "message": "User Id or password is invalid." }`
			
			`{ "status": "invalid", "message": "A <X> value is required for this type." }`
			
			`{ "status": "invalid", "message": "Unknown value. Supported values are: <X>." }`

kba Validate knowledge-based answer	`{ "user_id": "<USERNAME>", "type": "kba", "token": "<ANSWER>", "factor_id": "<KBQ PROPERTY>" }` **Example:** `{ "user_id": "jbeam", "type": "kba", "token": "biking", "factor_id": "KBQ2" }`	`{ "status": "valid", "message": "" }`	`{ "status": "invalid", "message": "Knowledge base answer is incorrect." }`
			`{ "status": "invalid", "message": "KBQ Id is out of range." }`
			`{ "status": "invalid", "message": "A <X> value is required for this type." }`
			`{ "status": "invalid", "message": "Unknown value. Supported values are: <X>." }`
oath Validate OATH token	`{ "user_id": "<USERNAME>", "type": "oath", "token": "<OTP>", "factor_id": "<DEVICE IDENTIFIER>" }` **Example:** `{ "user_id": "jbeam", "type": "oath", "token": "123456", "factor_id": "a0b1cd23e4f5g67h8ij90k" }`	`{ "status": "valid", "message": "" }`	`{ "status": "invalid", "message": "OTP is invalid." }`
			`{ "status": "invalid", "message": "A <X> value is required for this type." }`
			`{ "status": "invalid", "message": "Unknown value. Supported values are: <X>." }`
call Deliver OTP via phone call	`{ "user_id": "<USERNAME>", "type": "call", "factor_id": "<PHONE PROPERTY>" }` **Example:** `{ "user_id": "jbeam", "type": "call", "factor_id": "Phone1" }`	`{ "status": "valid", "message": "", "user_id": "jbeam", "otp": "8430" }`	`{ "status": "invalid", "message": "Request validation failed with: Unknown factor id '<X>'" }` See **Server Error** information below

sms
 	
Deliver OTP via text message
	`{`
	`    "user_id":`
	`"<USERNAME>",`
	`    "type": "sms",`
	`    "factor_id":`
	`"<PHONE PROPERTY>"`
	`}`
	
	Example:
	
	`{`
	`    "user_id": "jbeam",`
	`    "type": "sms",`
	`    "factor_id": "Phone1"`
	`}`
		`{`
		`    "status":`
		`"valid",`
		`    "message": "",`
		`    "user_id":`
		`"jbeam",`
		`    "otp": "8430"`
		`}`

email
 	
Deliver OTP via email
	`{`
	`    "user_id":`
	`"<USERNAME>",`
	`    "type": "email",`
	`    "factor_id":`
	`"<EMAIL PROPERTY>"`
	`}`
	
	Example:
	
	`{`
	`    "user_id": "jbeam",`
	`    "type": "email",`
	`    "factor_id": "Email1"`
	`}`
		`{`
		`    "status":`
		`"valid",`
		`    "message": "",`
		`    "user_id":`
		`"jbeam",`
		`    "otp": "8430"`
		`}`

push
 	
Deliver OTP via PUSH Notification
	`{`
	`    "user_id":`
	`"<USERNAME>",`
	`    "type": "push",`
	`    "factor_id":`
	`"<DEVICE IDENTIFIER>"`
	`}`
	
	Example:
	
	`{`
	`    "user_id": "jbeam",`
	`    "type": "push",`
	`    "factor_id":`
	`"z0y9x87wv6u5t43srq2p1on"`
	`}`
		`{`
		`    "status":`
		`"valid",`
		`    "message": "",`
		`    "user_id":`
		`"jbeam",`
		`    "otp": "8430"`
		`}`

help_desk
 	
Deliver OTP via help desk
	`{`
	`    "user_id":`
	`"<USERNAME>",`
	`    "type": "help_desk",`
	`    "factor_id":`
	`"<HELPDESK PROPERTY>"`
	`}`
	
	Example:
	
	`{`
	`    "user_id": "jbeam",`
	`    "type": "help_desk",`
	`    "factor_id": "HelpDesk1"`
	`}`
		`{`
		`    "status":`
		`"valid",`
		`    "message": "",`
		`    "user_id":`
		`"jbeam",`
		`    "otp": "8430"`
		`}`

/ipeval POST Endpoint

Endpoint: **https://SecureAuthIdPFQDN/SecureAuthIdPRealm/api/v1/ipeval**

For example: **https://secureauth.company.com/secureauth2/api/v1/ipeval**

The **/ipeval POST** endpoint enables SecureAuth IdP to evaluate the 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	JSON Parameters	Success Response	Failure / Error Response
risk IP Address Risk Evaluation	`{ "user_id": "<USERNAME>", "type": "risk", "ip_address": "<IP ADDRESS>" }` **Example:** `{ "user_id": "jbeam", "type": "risk", "ip_address": "11.222.33.44" }`	`{ "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": "Service is offline. IP could not be evaluated at this time." }` This response may occur because the SecureAuth IdP appliance does not have the required license for this feature. Contact **SecureAuth Support** to upgrade
			`{ "status": "invalid", "message": "Unknown value. Supported values are: risk." }`
			`{ "status": "invalid", "message": "<X> was not present in request." }`
			`{ "status": "invalid", "message": "Request validation failed with: Invalid IP address." }`
			See **Server Error** information below

Risk Factor (threatType) Scores

Threat Type (AE. IP.threatType)	Score	SecureAuth IdP Risk Category	Definition
Anonymous Proxy	100	Extreme	Authentication is coming from a server that is designed to hide or anonymize the actual source IP Address
Attacker	99	Extreme	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
Compromised	98	Extreme	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
Related	88	High	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
Victim	89	High	Indicators representing an entity that has been confirmed to have been victimized by malicious activity, where actors have attempted or succeeded compromise
Uncategorized	80	High	Uncategorized threat

threatCategory Scores

Threat Category (AE.IP. threatCategory)	**Response Value**	**Definition**
Anonymous Proxy	0	Authentication is coming from a server that is designed to hide of anonymize the actual source IP Address
Cyber Espionage	1	Global issue with highly sophisticated nation-states and other actors targeting military, political, and commercial interests to gain decision advantage
Hacktivism	2	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
Enterprise	3	Threats specifically targeted at Enterprise
Critical Infrastructure	4	Threats specifically targeted at Critical Infrastructure
Cyber Crime	5	Threats typically orchestrated by criminal elements for financial benefit
Vulnerability and Exploitation	6	Threats targeting known software vulnerabilities

If a server error is encountered, then the follow response is returned:

```
{
  "status": "server_error",
  "message": "<Exception message describing the issue.>",
}
HTTP Status 500
```