Contents

1. Introduction ............................................................................................................. 4
2. Integration Overview ................................................................................................. 4
3. Configuration .............................................................................................................. 4
4. Authentication API ..................................................................................................... 5
   4.1 Step 1: Start 2-Factor Authentication ................................................................. 5
      4.1.1 Request ........................................................................................................... 5
      4.1.2 Response ........................................................................................................ 5
   4.2 Step 2: Authenticate .............................................................................................. 6
      4.2.1 Request ........................................................................................................... 6
      4.2.2 Response ........................................................................................................ 6
5. User Management API ............................................................................................... 7
   5.1 Get User Profile ................................................................................................... 7
      5.1.1 Request ........................................................................................................... 7
      5.1.2 Response ........................................................................................................ 7
   5.2 Unlock ................................................................................................................... 8
      5.2.1 Request ........................................................................................................... 8
      5.2.2 Response ........................................................................................................ 8
   5.3 Deprovision .......................................................................................................... 9
      5.3.1 Request ........................................................................................................... 9
      5.3.2 Response ........................................................................................................ 9
   5.4 Provision Mobile Application ............................................................................... 9
      5.4.1 Request ........................................................................................................... 9
      5.4.2 Response ........................................................................................................ 10
   5.5 Provision Text Message ..................................................................................... 10
      5.5.1 Request ........................................................................................................... 10
      5.5.2 Response ........................................................................................................ 10
6. Error Handling .......................................................................................................... 11
   6.1 API Errors ............................................................................................................ 11
   6.2 HTTP Errors ....................................................................................................... 11
1. Introduction

In most web-based applications, users are authenticated before being granted access to protected resources. By asking for an additional authentication factor during the logon process, such applications gain an additional layer of security.

The ESET Secure Authentication API is a REST-based web service that can be used to easily add two-factor authentication (2FA) to existing applications.

2. Integration Overview

The API consists of two endpoints:

1. The Authentication API, for adding 2FA to existing applications.
2. The User Management API, for managing 2FA users.

The API operates using methods which are called by POSTing JSON-formatted text to the relevant API URLs. All responses are also encoded as JSON-formatted text containing the method result and any applicable error messages.

The API is available on all servers where the Authentication Core component is installed and runs over the secure HTTPS protocol on port 8001. The API is a subcomponent of the standard ESA Authentication Service. As such, an Active Directory domain and a functional ESA installation are prerequisites to using the API. Only Active Directory is supported, other user stores can not be used.

3. Configuration

The API is disabled by default and must be enabled before use. Each set of API credentials can be enabled for the Authentication API, the User Management API or both endpoints. Once enabled, API credentials must be created to authorize requests:

1. Launch the ESET Secure Authentication Management Console and navigate to the Advanced Settings node for your domain.
2. Expand the API section.
3. Select the check box next to API is enabled and click Save to save changes.
4. In the ESET Secure Authentication Management Console, navigate to the newly visible API Credentials node for your domain.
5. Click Add Credentials to create a new set of credentials.
6. Double-click the newly created credentials to get the username and password that are to be used for API authentication. Enable the credentials for the endpoint(s) that are required:
   a. If the credentials are to be used for the Authentication API, then tick the Enabled for Auth API check box.
   b. If the credentials are to be used for the User Management API, then tick the Enabled for User Management API check box.
   c. Click OK to save the changes.
7. Press the Windows key + R, type Services.msc into the Open field and then press Enter to open the Windows Services Console.
8. Right-click ESET Secure Authentication Core and select Restart from the context menu.

You can create several sets of API credentials. We recommend that you create different sets for each application being protected, as well as a set for testing.
If the API is enabled, all servers with the Authentication Core component installed will respond to authorized API requests after they are restarted. The Authentication Core service must be restarted when credentials are created or deleted.

4. Authentication API

All Authentication API methods are available on URLs of the form `https://127.0.0.1:8001/auth/v1/` and are protected from unauthorized access via standard HTTP Basic Authentication, requiring a valid set of API Credentials that are enabled for the Authentication API before processing any request. The `Content-Type` header must be set to `application/json` for each request.

The ESET Secure Authentication installer automatically uses an appropriate SSL security certificate installed on the machine, or generates a new self-signed certificate if another cannot be found.

Replacing the SSL certificate is covered in the ESA API SSL Certificate replacement document.

4.1 Step 1: Start 2-Factor Authentication

As soon as the existing application has verified a user’s username and password, the Start 2-FA method must be called in order to determine whether two-factor authentication has been enabled for the user. If required, an SMS OTP will automatically be sent to the user at this time.

4.1.1 Request

To begin the 2FA process, make an HTTP POST request to the following URI:

/auth/v1/start2fa

The following JSON string must be posted:

```json
{
    "username": "USERNAME"
}
```

The `username` field is a string with the `samAccountName` of the user to be authenticated. It is very important that the correct username be sent to the API: the `samAccountName` is the user’s normal logon name in Active Directory.

4.1.2 Response

All typical responses will be returned with a 200 (OK) HTTP status code, even if the requested action failed. The response will be a JSON string. See below for an example of a standard response:

```json
{
    "expected_otp": ["APP", "SMS"],
    "error": "ERROR_NONE",
    "error_message":"
}
```

If no error has occurred, then the `error` field will display “ERROR_NONE”. Please see the Error Handling section for a description of possible error codes.

The `error_message` field will give a friendly description of the error, if applicable.
The **expected_otp** field is an array and specifies the OTP (One-time password) types that can be expected from the user. This value can assist with UI creation, for example, it will indicate if the user should expect an SMS or not. If the array is empty then no OTP is required (i.e. 2FA is not enabled) and the user should be logged in immediately. The following OTP types can be included in the array:

- **APP** – the user has already installed the ESA application on their mobile phone and should generate an OTP using the application.
- **SMS** – the user has not installed the application and has been sent an SMS with an OTP.
- **HARD_TOKEN** – the user has been assigned a hard token and should generate an OTP using the device.

### 4.2 Step 2: Authenticate

#### 4.2.1 Request

To authenticate a user, make an HTTP POST request to the following URI:

```
/auth/v1/authenticate
```

The following JSON string must be posted:

```json
{
    "username": "USERNAME",
    "otp": "123456"
}
```

The **username** field is a string with the `samAccountName` of the user to be authenticated and the **otp** field a string with the OTP entered by the user.

#### 4.2.2 Response

All typical responses will be returned with a 200 (OK) HTTP status code, even if the requested action failed. The response will be a JSON string. See below for an example of a standard response:

```json
{
    "authenticated": true,
    "error": "ERROR_NONE",
    "error_message": ""
}
```

If no error has occurred, then the **error** field will display **ERROR_NONE**. Please see the Error Handling section of this guide for a description of possible error codes.

The **error_message** field will give a description of the error if an error has occurred.

The **authenticated** field is a Boolean that specifies whether the supplied OTP is valid. If the **authenticated** value is **true**, the user’s OTP has been successfully validated and the user should be logged in.
5. User Management API

All User Management API methods are available on URLs of the form https://127.0.0.1:8001/manage/users/v1/ and are protected from unauthorized access via standard HTTP Basic Authentication, requiring a valid set of API Credentials that are enabled for the User Management API before processing any request. The Content-Type header must be set to application/json for each request.

The ESET Secure Authentication installer automatically uses an appropriate SSL security certificate installed on the machine, or generates a new self-signed certificate if another cannot be found.

Replacing the SSL certificate is covered in the ESA API SSL Certificate replacement document.

5.1 Get User Profile

This method returns 2FA information about a user account.

5.1.1 Request

To get the 2FA profile of a user, make an HTTP GET request to the following URI:

/manage/users/v1/profile/USERNAME

Where USERNAME is a string with the samAccountName of the user to fetch the profile of. It is very important that the correct username be sent to the API: the samAccountName is the user’s normal logon name in Active Directory. The username must be URL-encoded.

5.1.2 Response

All typical responses will be returned with a 200 (OK) HTTP status code, even if the requested action failed. The response will be a JSON string. See below for an example of a standard response:

```
{
    "username": "USERNAME",
    "mobile_number": "2700000",
    "is_locked": false,
    "last_success": "2014-01-01T00:00:00",
    "last_failure": null,
    "consecutive_failures": 0,
    "credential_type": ["APP", "SMS"],
    "error": "ERROR_NONE",
    "error_message": ""
}
```

If no error has occurred, then the error field will display ERROR_NONE. Please see the Error Handling section of this guide for a description of possible error codes.

The error_message field will give a description of the error if an error has occurred.

The username field is a String containing the samAccountName of the user.

The mobile_number field is a String containing the mobile number of the user.

The is_locked field is a Boolean that specifies if the user has been locked for 2FA due to too many failed authentication attempts.
The last_success field is a Date that specifies the last time that the user performed a successful authentication. This field can be null.

The last_failure field is a Date that specifies the last time that the user performed a failed authentication. This field can be null.

The consecutive_failures field is an Integer that specifies the number of consecutive failed authentication attempts performed by the user.

The credential_type field is an array and specifies the OTP (One-time password) types that have been enabled for the user. The following OTP types can be included in the array:

- **APP** – the user has been enabled for the ESA Mobile App.
- **SMS** – the user has been enabled for SMS OTPs.
- **HARD_TOKEN** – the user has been enabled for hard token OTPs.

## 5.2 Unlock

This method will unlock the 2FA access of a user. It will not unlock an account locked by Active Directory.

### 5.2.1 Request

To unlock a user, make an HTTP POST request to the following URI:

```
/manage/users/v1/unlock
```

The following JSON string must be posted:

```json
{
    "username": "USERNAME"
}
```

The username field is a string with the `samAccountName` of the user to unlock. It is very important that the correct username be sent to the API: the `samAccountName` is the user’s normal logon name in Active Directory.

### 5.2.2 Response

All typical responses will be returned with a 200 (OK) HTTP status code, even if the requested action failed. The response will be a JSON string. The response will only contain a possible error code and message, without any other data. See below for an example of a standard response:

```json
{
    "error": "ERROR_NONE",
    "error_message": ""
}
```

If no error has occurred, then the error field will display **ERROR_NONE**. Please see the Error Handling section of this guide for a description of possible error codes.

The error_message field will give a description of the error if an error has occurred.
5.3 Deprovision

This method will disable 2FA for a user.

5.3.1 Request

To disable 2FA for a user, make an HTTP POST request to the following URI:

```
/manage/users/v1/deprovision
```

The following JSON string must be posted:

```
{
    "username": "USERNAME"
}
```

The `username` field is a string with the `samAccountName` of the user to disable 2FA for. It is very important that the correct username be sent to the API: the `samAccountName` is the user’s normal logon name in Active Directory.

5.3.2 Response

All typical responses will be returned with a 200 (OK) HTTP status code, even if the requested action failed. The response will be a JSON string. The response will only contain a possible error code and message, without any other data. See below for an example of a standard response:

```
{
    "error": "ERROR_NONE",
    "error_message": ""
}
```

If no error has occurred, then the `error` field will display `ERROR_NONE`. Please see the Error Handling section of this guide for a description of possible error codes.

The `error_message` field will give a description of the error if an error has occurred.

5.4 Provision Mobile Application

This method will enable a user for Mobile Application OTPs. A text message with the installation URL for the mobile application will be sent to the user.

5.4.1 Request

To provision a user for the Mobile Application, make an HTTP POST request to the following URI:

```
/manage/users/v1/provisionmobileapp
```

The following JSON string must be posted:

```
{
    "username": "USERNAME"
}
```

The `username` field is a string with the `samAccountName` of the user to provision. It is very important that the correct username be sent to the API: the `samAccountName` is the user’s normal logon name in Active Directory.
5.4.2 Response
All typical responses will be returned with a 200 (OK) HTTP status code, even if the requested action failed. The response will be a JSON string. See below for an example of a standard response:

```json
{
    "installation_url": "http://...",
    "error": "ERROR_NONE",
    "error_message": ""
}
```

If no error has occurred, then the **error** field will display **ERROR_NONE**. Please see the Error Handling section of this guide for a description of possible error codes.

The **error_message** field will give a description of the error if an error has occurred.

The **installation_url** field is a String that contains the installation URL for the Mobile Application.

5.5 Provision Text Message
This method will enable a user for text message OTPs.

5.5.1 Request
To provision a user for the text message OTPs, make an HTTP POST request to the following URI:

```
/manage/users/v1/provisiontextmessage
```

The following JSON string must be posted:

```json
{
    "username": "USERNAME"
}
```

The **username** field is a string with the **samAccountName** of the user to provision. It is very important that the correct username be sent to the API: the **samAccountName** is the user’s normal logon name in Active Directory.

5.5.2 Response
All typical responses will be returned with a 200 (OK) HTTP status code, even if the requested action failed. The response will be a JSON string. The response will only contain a possible error code and message, without any other data. See below for an example of a standard response:

```json
{
    "error": "ERROR_NONE",
    "error_message": ""
}
```

If no error has occurred, then the **error** field will display **ERROR_NONE**. Please see the Error Handling section of this guide for a description of possible error codes.

The **error_message** field will give a description of the error if an error has occurred.
6. Error Handling

6.1 API Errors

All API errors will be returned as a response with an HTTP 200 (OK) status code.

The error field in the JSON response will indicate the error code, which is a literal string value. The following error codes are defined:

- ERROR_NONE: No error has occurred
- ERROR_USER_NOT_FOUND: The supplied username does not exist in the system
- ERROR_FAULT: An unspecified error has occurred

In addition to the error field, an error_message is also provided with a friendly description of the error. Only the error field should be used to determine error conditions as the error_message field is only informational and is subject to change without notice.

6.2 HTTP Errors

All HTTP errors will be returned as responses with an empty body and an HTTP status code other than the normal 200 (OK).

The following erroneous HTTP status code can be returned:

- HTTP 500 (Internal Server Error): The API service experienced an unknown, fatal error
- HTTP 400 (Bad Request): The format of the “Authorization” header in the HTTP request is invalid
- HTTP 401 (Unauthorized): No API credentials were supplied with the HTTP request
- HTTP 403 (Forbidden): Credentials supplied with the HTTP request are invalid.