VeriSaaS keymaker API v1

release 2020Q3

Release	Date	Description	Author	Reviewer	Approver
0.2.4	02/10/2020	keymaker SaaS API 2020Q3 release	IPA	GSS	MSY

1. Introduction

KeyMaker service provides a credential management tool for VeriSaaS users. It allows to do the following tasks::

- Check and rotate (create new and delete old ones) their API keys
- Check and modify their allowed IP source list where the service can be consumed from

1.1 API key credentials

VeriSaaS API consumers are authenticated using API Key credentials provided by Veridas which should be sent as a header in every request enabling users to consume the services' functionalities.

Keymaker allows the users of these services to create new API Key credentials and also to revoke API Keys which no longer want to be used. To do that, certain requests have to be made to the Keymaker service, providing the required API key information that the user wants to register or to remove.

Rotating the API Keys on a regular basis is a good security practice. It is recommended to re-create/rotate them at least once per year.

To keep your API keys secure, try to follow these best practices:

- Do not embed API keys directly in the code of your frontend application
- Delete unused API keys
- Rotate your API keys periodically
- Rotate API keys which may be potentially compromised
- Do not store API keys on any source control repository

1.2 IP Source Allow List

Another security mechanism implemented on VeriSaaS is IP access restriction.

All requests coming from an unregistered source IP will be automatically rejected (403 code) by VeriSaaS, so it is necessary to have an IP source allow list with at least one IP.

Keymaker allows you to manage that list for every service independently.
Some best practices to maintain IP Source allow list are the following:

- Review periodically which IPs are configured
- Delete unused/unnecessary IPs

2. Changelog

2.1. Added

- Added support for IP Allow List Management

3. Operations

3.1 API key Management

Keymaker Key management API allows you to obtain the API key list, add new API keys to the list and remove API keys from it.

The most common operations to carry out with this API are to rotate an API Key and to Check the API keys age.

3.1.1 How to rotate an API key

1. Create a new API Key
2. Update the APIKey on the code of the Veridas application integration to start using the new one. Both the old and new keys are functional at this point so the applications will not be disrupted during this process
3. Verify that the old API key is not used by any other applications, servers or platforms and then remove it from the Keymaker list
4. Check that only the necessary API keys are in the list

3.1.2 How to check API Keys age

1. Retrieve all available API keys by using the Keymaker API
2. Review “created_at” field and verify the age of that API Key
3. If you consider that the age is old enough for rotation, then please follow the steps in the previous section.

There is no hard limit for the age of an APIkey, but Veridas recommends not to exceed one year for security reasons.

3.2 IP Allow list Management

Keymaker IP management API allows you to obtain the configured IP allow list and to update it by adding or removing IPs from the list.
The IP addresses have to be defined and provided to the API in CIDR notation. For more information about this, we recommend the article https://www.ipaddressguide.com/cidr.

There are some restrictions in the IP addresses that can be added by the users:

- Must not belong to a private subnet (ie: 192.168.0.0/16)
- Must not belong to a reserved subnet (ie: 224.x.x.x)
- The mask used cannot be less than /16 (ie: /12, /8, etc)
- IPv6 is not supported

The most common operations to carry out with this API are to add a new IP address to the allow list and to review the allow list content.

3.2.1 How to add a new IP address to the Allow List

1. Check that the IP address that wants to be added is not already added to the list
2. Replace the *allow list* by adding the new IP address
3. Check that the IP address exists into the *allow list* and verify that the service can be accessed by using it

3.2.2 How to do a periodical Allow List revision

1. Obtain the allow list configured in every service available
2. Review if all IP addresses are necessary
3. If you consider that one or more of addresses are no longer necessary, you should replace the *allow list* with a new one where that addresses got removed.
4. Verify that only the necessary addresses are configured

4. API Considerations

The following are some general considerations about this API that must be taken into account before consuming the service.

4.1. Authentication

The authentication method is API key based
All requests must include the **apikey** authorization header provided by Veridas for using its services.

Should you have more than one Verisaas cloud account, please ensure that you use the appropriate credentials for the service you are looking to manage their authentication

APIKEYs or IP allow list. For instance, you cannot use credentials from vali-Das service to manage the authentication APIKeys or IP allow list of das-Peak service.

Also, ensure that every query to keymaker service is sent from a registered IP address on the cloud account you are trying to manage.

4.2. Requests

- The content-type header must be used on every request with the “json” value
- The API is HTTP-based and uses TLS everywhere with valid certificates. For security reasons, customers should never trust VeriSaaS endpoints exposing invalid certificates
- The service includes an `/alive` endpoint that returns the 204 HTTP status code if the service is up and running. This can be used to check the service’s health

All responses will be encoded using JSON. Responses will return a suitable HTTP status code indicating if the request was successful (200/201 or 204 if nothing else is returned) or not (any other code). Responses will also include a code field in the JSON body that can provide more information about the specific error on each case.

In general, successful responses will have the following formats:

HTTP Status: 200 OK

```
{
	“items”: [
		{
			{"field": "value1"},
			{"field": "value2"}
		}
	]
}
```

or

HTTP Status: 200 OK

```
{
	"field": "value1",
	"field": "value2"
}
```

or

HTTP Status: 204 NO CONTENT

where:

Field	Required	Description
items	no	Message payload. It will contain a JSON content defined by each endpoint.

In case of error:

Field	Required	Description
code	yes	Error code
message	no	A message indicating what went wrong

Example:

```
{
      “code”: “InvalidCredentials”,
      “message”: “Invalid authentication credentials”
}
```

4.3 Versioning

The API version will be included in the URL, after the base url and before the endpoint:

`https://<base_url>/keymaker/`**`v{number:integer}`**`/<endpoint>`

Non-backward compatible changes will cause a version increment. As of now, the API only supports the **v1** version.

i.e. for the sandbox environment:

`GET https://api-work.eu.veri-das.com/keymaker/`**`v1`**`/alive`

5. API Definition

The following endpoints are exposed:

Public Base URL (v1):

`https://<base_url>/keymaker/v1/`

Resources:

Method	Public URL	Description
`GET`	`/alive`	Checks if the service is up
`POST`	`/keys`	Creates a new API key
`GET`	`/keys`	Gets a list of API keys
`GET`	`/keys/{id}`	Retrieves an API key
`DELETE`	`/keys/{id}`	Deletes an API key
`GET`	`/ip-allowlist`	Gets the Allow List configured
`PUT`	`/ip-allowlist`	Update IP Allow List

5.1. Check if the service is alive

`GET /alive`

Response

HTTP Status: 204 NO CONTENT

5.2. Create a new API key

Creates a new API key. Returns the newly created API key information.

Note: There is a limit on the number of API keys a user can have at any given time. The maximum configured is 10. if the client has already reached their API key number limit, an error code will be returned.

`POST /keys`

Response

HTTP Status: 201 CREATED

Returns the API key id, API key, created date and profile.

{

```
        “id”: “API_KEY_ID”,
        “key: “API_KEY”,
        “created_at”: created date (utc format),
        “profile”: “default”
}
```

Errors:

Code	HTTP Status	Message
TooManyApiKeysError	400	You can not create more API keys. Please delete some of the existing ones first.

5.3. Get API keys

Retrieves a list of API keys.

`GET /keys`

Response

HTTP Status: 200 OK

```
{
        “items”: [
                {
                        “id”: “API_KEY_ID”,
                        “key: “API_KEY”,
                        “created_at”: created date (utc format),
                        “profile”: “default”
                },
                {
                        “id”: “API_KEY_ID”,
                        “key: “API_KEY”,
                        “created_at”: created date (utc format),
                        “profile”: “default”
                },
                …
        ]

}
```

5.4. Get API key by id

Retrieve an API key.

`GET /keys/{api_key_id}`

Response

HTTP Status: 200 OK

```
{
      “items”:
            {
                  “id”: “API_KEY_ID”,
                  “key: “API_KEY”,
                  “created_at”: created date (utc format),
                  “profile”: “default”
            }
}
```

Errors:

Code	HTTP Status	Message
ApiKeyNotFoundError	404	We could not find the API key you were looking for

5.5. Delete an API key

Deletes an API key.

Note: For security reasons, it is not possible to delete the same API key that the client used in the header to authenticate the request.

`DELETE /keys/{api_key_id}`

Response

HTTP Status: 204 NO CONTENT

Errors:

Code	HTTP Status	Message

ApiKeyDeleteError	400	This API key can not be deleted. Please ensure you are not trying to delete the key currently being used
ApiKeyNotFoundError	404	We could not find the API key you were looking for

5.6 Get IP Allow List

Retrieves the IP Allowlist

`GET /ip-allowlist`

Response

HTTP Status: 200 OK

```
{
      “addresses”: ["208.130.29.33/32", "201.23.14.0/30"]
}
```

5.7 Update IP Allow List

Adds/deletes one or more addresses. Returns the new IP allow list configured.

Note: There is a limit on the number of IP addresses that a user can have at any given time. This maximum is set as 20. if the user has already reached its IP addresses number limit, an error code will be returned.

`PUT /ip-allowlist`

Request

Name	Required	Type	Description
addresses	yes	json	Array with the list of address wants to configure

Response

HTTP Status: 200 OK

```
{
        “addresses”: ["208.130.29.33/32", "201.23.14.0/30"]
}
```

Errors:

Code	HTTP Status	Message
TooManyAddressesError	400	You exceeded the maximum addresses configured.
ValidationError	400	The provided data is not valid.

6. Generic API errors

Errors:

Code	HTTP Status	Message
IPRestrictionNotConfigured	404	IP-Restriction service is not currently enabled*
InvalidCredentials	401	Invalid authentication credentials
UnexpectedError	500	An error occurred while processing your request. Please try again and if the problem persists contact the system administrator

*Please contact Veridas Support Desk for its activation.