# SAFE 3 integration manual – SSO

### 

# Information on SAFE Clients

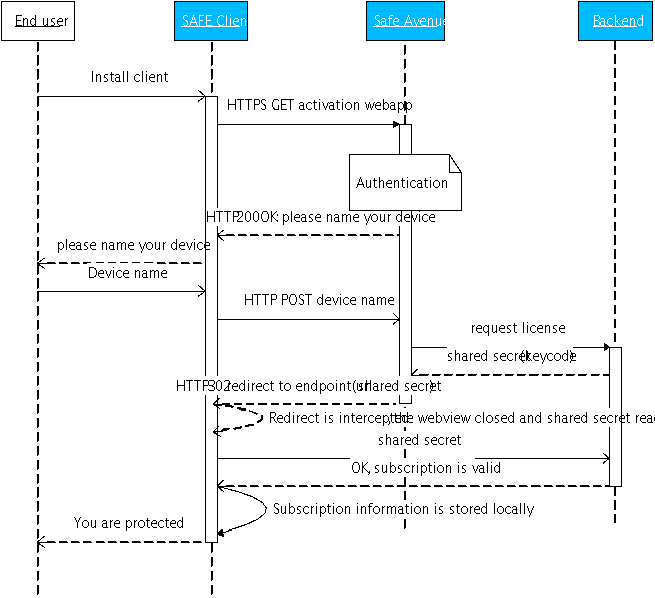
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| C:\cae181b73ab1109b5ed1a9db94c6e5d2 | C:\1dd157838fa99145ce9940cbe153fb74 |  |  |  |

The SAFE Clients are the applications that provide the actual protection of the end users devices. Due to differences between operating systems, the exact functionality provided differs slightly per platform. Regardless, the user experience of the SAFE Clients is designed to be as coherent as possible.

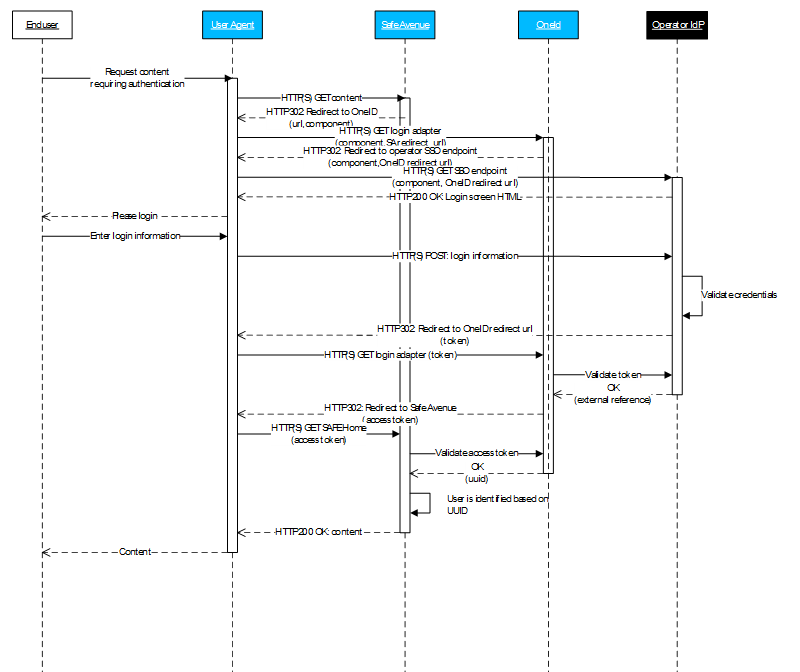
### Activation user journey

To activate a SAFE Client, users are requested to authenticate and assign a name for the device they are protecting:  

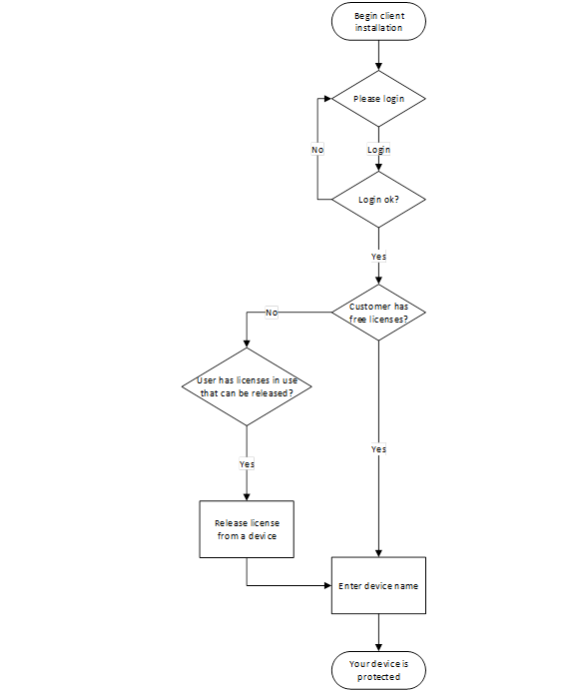

#### ****SAFE Client Activation sequence diagram****

The communication between components related to the best case scenario, where the user already had valid credentials and free licenses, described as a sequence diagram:   
  
The result of this sequence is that the client receives a shared secret (subscription key), which is used to authorize future communication between the backend and the SAFE Client. If the SAFE account itself is terminated or expired, or if the user decides to release the license used on the device, Safe Avenue terminates the license from the backend. Once the clients tries connecting to the backend, it will then receive the information that the license has been terminated.  
This means that the "logged in" state is not stored in the client, and the users do not need to login to the client again to enable normal functionality. The login is only required during initial activation.

The actual login sequence in an SSO integration usually closely follows the following sequence:



#### ****SAFE Client Activation business logic****

The user experience sequence provided by the SAFE Client activation process described as a flow chart:  
   
During the SAFE Client activation process, there are several potential situations that might require user interaction before we can allow the activation to proceed. Examples include:

* The user does not have an account
* The user has an account but has forgotten the password
* The user has an account but does not have free licenses

The SAFE Activation Web App embedded in the clients includes the logic as detailed above to allow the user to resolve these situations and finish the installation.

#### SAFE Client Activation content sources

During the client activation, the user is seeing content from several sources. The exact installation flow varies slightly depending on the platform on which is used to install, the example provided here is from the SAFE Windows PC Client installation flow.

1. Login

|  |  |
| --- | --- |
| C:\7575790ee5bb4079432a75ea77f234dd | After the user starts the SAFE Windows client installation, the installer opens and asks the user to log in to proceed.  The implementation works by opening a webview within the application, which loads the login screen via HTTPS.  With an SSO integration, the login screen html is coming from the Operator WWW –servers. |

1. SAFE Business logic

|  |  |
| --- | --- |
| C:\9a35e9daa6c1e471386fdf4b70c3b514 | After successful login, the SAFE business logic is performed as detailed in the previous chapter.  The SAFE business logic content is dynamically loaded from the F-Secure www infrastructure using a browser element within the client application |

1. SAFE Client setup

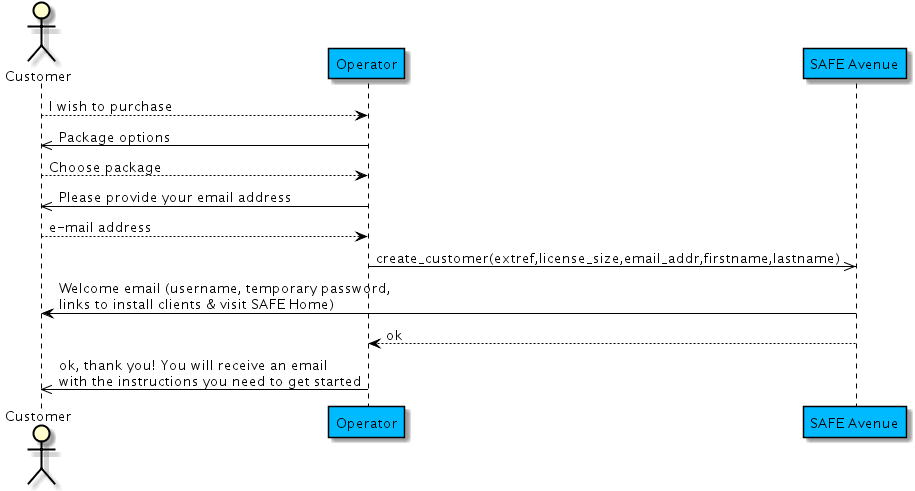
The actual order of the steps vary depending on the platform. Any additional content is coming from the SAFE Client (or the installer of the SAFE client.)  


# User Experience integration – Use cases

## PROVISIONING use cases

### UCP-1: A Customer purchases SAFE from the Operator

SYNOPSIS: A customer purchases SAFE via any available channel – be it a walk in store, the Operator's web shop, SMS, outbound call center or any other means.   
EXPECTED END RESULT: The Customer has a SAFE account created and linked to his or her operator credentials, and is able to start using the service.  
  
**Sequence diagram:**



**Involved components:**

* Operator sales channels
* SAFE Avenue API
* Optional: SAFE Welcome email

**Integration methods:**

* Create\_customer()

For this scenario, the mandatory parameters are:

* + Extref: reference for the customer in the operator CRM. This reference is used in the future for all API methods to modify the Customer entity in SAFE Avenue.
  + License\_size: the number of devices the Customer is allowed to protect using SAFE.
  + Username: the username of the user. This value is normally used by the SSO integration to match the user logging in to the created account in F-Secure systems.
  + Firstname: the first name of the user. If the SAFE 3 Home UI is in use, the first names of the users are visible there.
  + Lastname: the last name of the user.

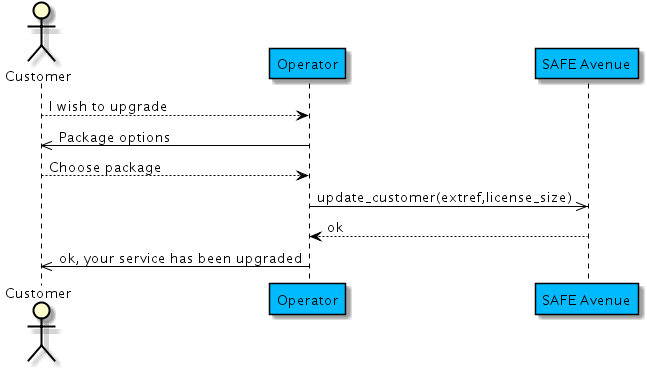
Optional parameters:

* + Locale: if the locale is given, the welcome email will be sent using the template configured for the given locale. If omitted, the welcome email will be sent using the template configured for the default locale of the Operator
  + Send\_email: if set to true (1), the welcome email will be sent by F-Secure.
  + CONTINUOUS

### UCP-4 A Customer upgrades the license size of a purchased subscription

SYNOPSIS: A customer who has a subscription upgrades the license size to protect additional devices.   
EXPECTED END RESULT: The customer's subscription is upgraded to the desired license size.

**Sequence diagram:**



**Involved components:**

* Operator sales channels
* SAFE Avenue API

**Integration methods:**

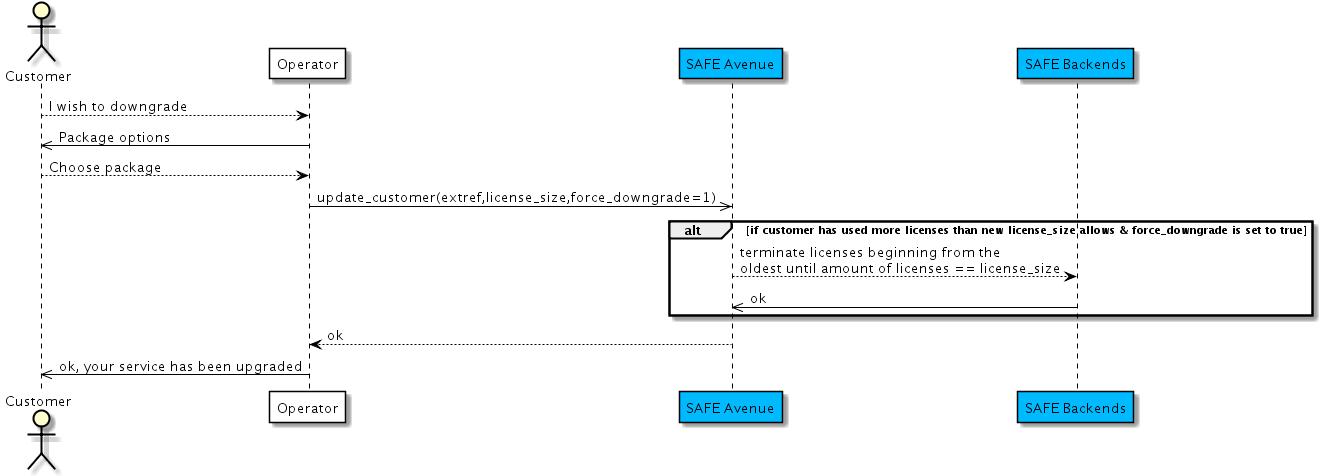
* Update\_customer()

For this scenario, the mandatory parameters are:

* + Extref: the external reference of the customer
  + License\_size: the license size matching with the service package purchased by the Customer

### UCP-5 A Customer downgrades the license size of a purchased subscription

SYNOPSIS: A customer who has a subscription downgrades the license size.   
EXPECTED END RESULT: The customer's subscription is downgraded to the desired license size.   
  
**Sequence diagram:**



**Involved components:**

* Operator sales channels
* SAFE Avenue API

**Integration methods:**

* Update\_customer()

For this scenario, the mandatory parameters are:

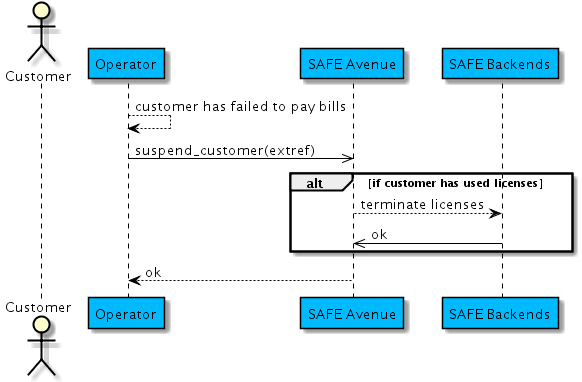
* + Extref: the external reference of the customer
  + License\_size: the license size matching with the service package purchased by the Customer

Optional parameters:

* + Force\_downgrade: if set to true (1), Safe Avenue will force the downgrade of the license\_size. If the customer had used more licenses than the new license\_size allows, Safe Avenue will terminate enough licenses to match the new license\_size, beginning from the oldest. If force\_downgrade is not given, Safe Avenue will return an error if the customer has used more licenses than the new license\_size would allow.

### UCP-6 A Customer's subscription is suspended (unpaid bills)

SYNOPSIS: A customer has for example failed to pay his bills, and his subscription is suspended.   
EXPECTED END RESULT: The customer's subscription is suspended.   
  
**Sequence diagram:**



**Involved components:**

* Operator CRM
* SAFE Avenue API

**Integration methods:**

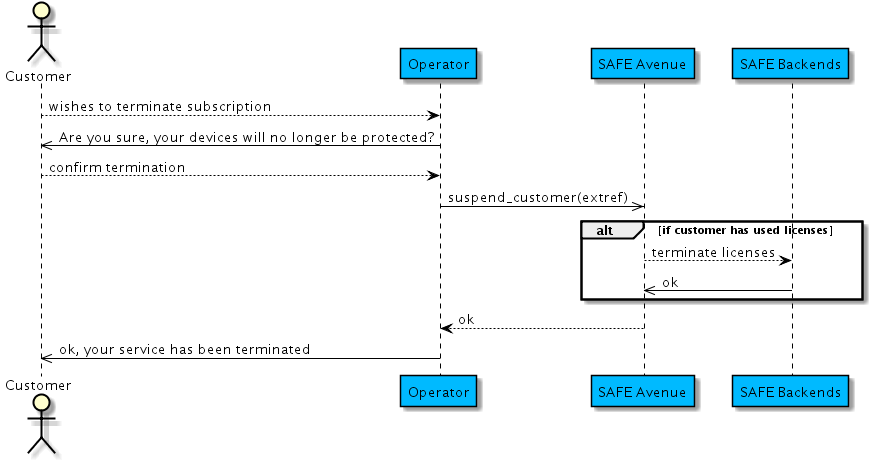
* suspend\_customer()

For this scenario, the mandatory parameter is:

* + Extref: the external reference of the customer

### UCP-7 A Customer decides to terminate his subscription

SYNOPSIS: A customer wants to terminate the SAFE subscription.   
EXPECTED END RESULT: The customer's SAFE subscription is terminated.   
  
**Sequence diagram:**



**Involved components:**

* Operator CRM
* SAFE Avenue API

**Integration methods:**

* suspend\_customer()

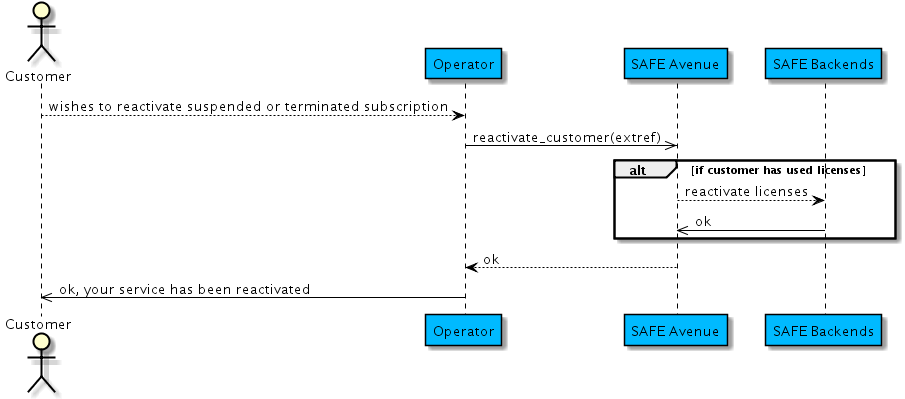
For this scenario, the mandatory parameter is:

* + Extref: the external reference of the customer

### UCP-8 A Customer's suspended or terminated subscription is reactivated

SYNOPSIS: A customer wants to reactivate a terminated or suspended SAFE subscription.   
EXPECTED END RESULT: The customer's SAFE subscription is terminated.

**Sequence diagram:**



**Involved components:**

* Operator CRM
* SAFE Avenue API

**Integration methods:**

* resume\_customer()

For this scenario, the mandatory parameter is:

* + Extref: the external reference of the customer

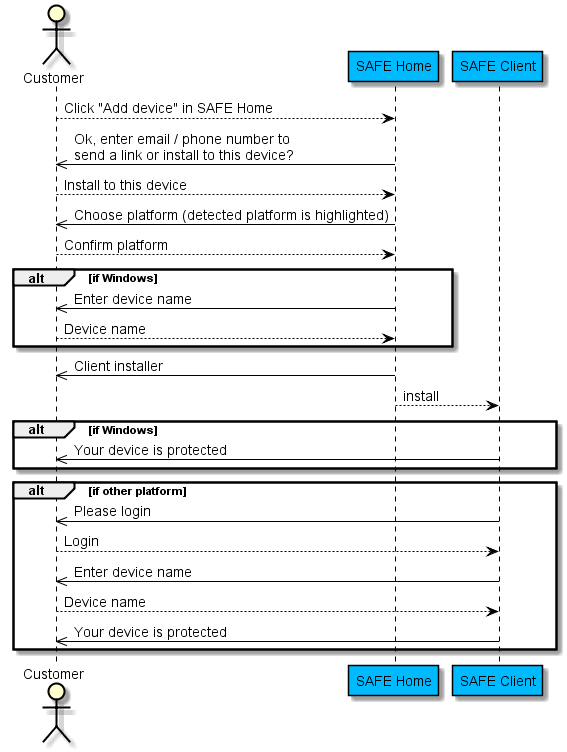
## Client installation use cases

The Client installation use cases do not require any API integration from the operator, however it is important for an Operator to review them, as most of them are linked to customizable and cobrandable component, and understanding these use cases is imperative to understand the SAFE user journeys from end to end.

UCI-1: User triggers SAFE Client installation from SAFE Home to the device that they are currently using

SYNOPSIS: A customer wants to protect the device that they are currently using to browse SAFE Home   
EXPECTED END RESULT: The customer's current device is protected

**Sequence diagram:**

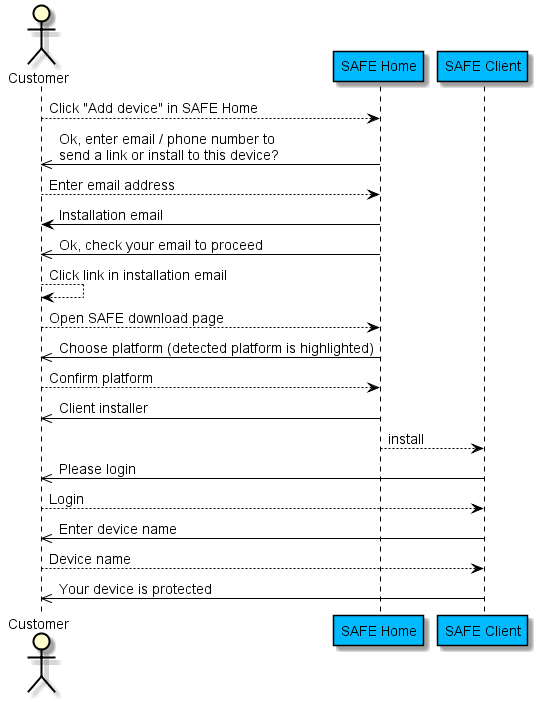
   
**Involved components:**

* SAFE Home
* SAFE Client

### UCI-2: User triggers SAFE Client installation from an installation email

SYNOPSIS: A customer wants to protect a device other than the one they are using to browse SAFE Home.   
EXPECTED END RESULT: The customer's device is protected.

**Sequence diagram:**

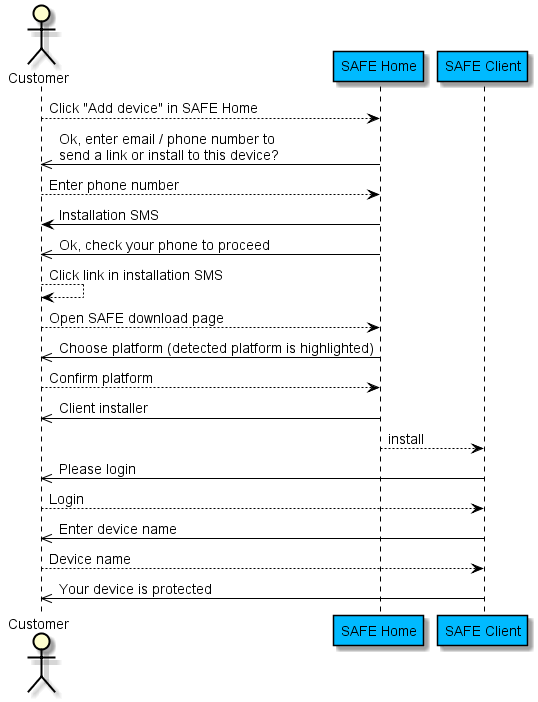
  
   
**Involved components:**

* SAFE Home
* SAFE Client
* SAFE Installation email

### UCI-3: User triggers SAFE Client installation from an installation SMS

SYNOPSIS: A customer wants to protect a device other than the one they are using to browse SAFE Home.   
EXPECTED END RESULT: The customer's device is protected.

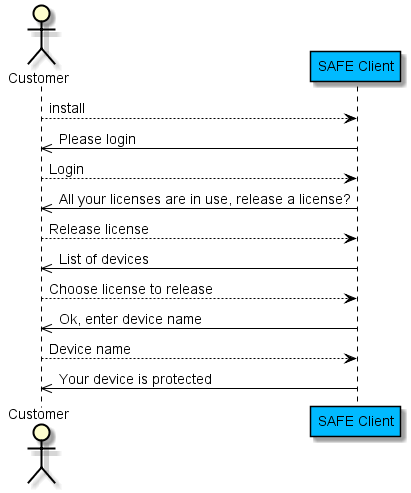
**Sequence diagram:**

  
   
**Involved components:**

* SAFE Home
* SAFE Client
* SAFE Installation SMS

### UCI-4: A Customer tries to install a SAFE Client, but all licenses are in use. The Customer frees a license to proceed

SYNOPSIS: A customer wants to protect a device and has already started the setup, but the customer has used up the entire license quota. The customer needs to release a license to proceed.   
EXPECTED END RESULT: The customer's device is protected.   
  
**Sequence diagram:**

  
   
**Involved components:**

* SAFE Home
* SAFE Client

# Full API documentation

Safe Avenue API is a JSON-RPC API that allows client to query and update customer data.

## Versioning and compatibility

Safe Avenue API version has the format {major}.{minor}, for example version 2.1.

Minor version update is backward compatible, so upgrading minor version should not break. For example, version 2.1 is backward compatible with version 2.0, upgrading from version 2.0 to 2.1 shouldn't need much (if any) changes. However, major version update **may not** be backward compatible. For example, upgrading from version 2.1 to 3.0 is expected to break and client should update accordingly.

To ensure backward compatibility on **minor releases**, F-Secure:

* **Will not** add mandatory parameters.
* **Will not** remove or rename response JSON objects.
* **Will not** remove or rename procedures.

F-Secure **may not** follow the above strategy on major releases.

Any new features requiring API or significant user interface changes will only be made available on the next minor/major version, whether the features are requested by partners or proactively developed by F- Secure. This means that an operator requesting a change to Safe Avenue, will have to move to the latest version of the API and Iframe to use the feature. Client product versions are attached to one particular Safe Avenue version, as the user experience may require changes to better support different client version.

To ensure that new features and updates are available on minor releases, F-Secure:

* **May** add optional parameters.
* **May** add items in response JSON objects.
* **May** add new procedures.

Because of this and to avoid frequent client updates, client must:

* **Not** expect a fixed number of items in response JSON objects.
* **Not** place a limit of response header and body length.

## URL and authentication

Safe Avenue API can be accessed from URL with the format:

https://{hostname}/api/{account}/{version}/{procedure}

|  |  |
| --- | --- |
| **hostname:** | Safe Avenue hostname, for example safeavenue.f-secure.com. |
| **account:** | Client account name, usually company name or its abbreviation without any space or symbol. |
| **version:** | API version, dotted version prefixed with v, for example v2.1. |
| **procedure:** | Procedure name. |

For example:

https://safeavenue.f-secure.com/api/helios/v3.0/echo?message=hi

Safe Avenue API can only be accessed via HTTPS, HTTP is not allowed. The certificates are included in common browsers by default, so Safe Avenue API works when accessed from browser. But when accessing the API from your application, remember to import appropriate root certificate if it is not imported by default. Only **root certificates** should be imported but **not intermediate or Safe Avenue certificates**. Safe Avenue certificates are updated without prior notice and intermediate certificates are subject to change. The root certificate can be obtained by exporting it from certificates manager of trusted browser, Firefox browser for example.

Safe Avenue API requires HTTP basic authentication, username and password must be of Operator Webservice account created by F-Secure SIE.

## GET vs POST

Safe Avenue API accepts GET requests on procedures that query data and POST requests on procedures that update data. Sending GET requests to a POST procedure or POST requests to a GET procedure will result in HTTP 405 error.

Parameters are passed to GET procedures in URL, for example:

GET https://safeavenue.f-secure.com/api/helios/v3.0/echo?message=hi

Parameters are passed to POST procedures in form data (application/x-www-form-urlencoded).

## Response

Safe Avenue API may response either success, general error or internal error. A success response means the request was received, accepted and processed successfully. A general error response means an error has occurred and client may interpret the error code to determine the root cause. An internal error response means unexpected error has occurred, it could be either an API bug or system downtime.

The response content may be in JSON or text. JSON content response has the header:

Content-Type: application/json; charset=utf-8

While text content response has one of the following headers:

* Content-Type: text/plain
* Content-Type: text/html

A success response has HTTP status 2xx, and with JSON content ok=true, data and error=null, for example:

{

"ok": true,

"data": {

"result": "Result",

"result2": "Result 2"

},

"error": null

}

A general error response has HTTP status of non-2xx, and with JSON content ok=false, data=null, error.code and error.message, for example:

{

"ok": false,

"data": null,

"error": {

"code": "does\_not\_exist",

"message": "Object does not exist"

}

}

An internal error response has HTTP status 5xx with JSON content ok=false and error.code="internal\_error", for example:

{

"ok": false,

"data": null,

"error": {

"code": "internal\_error",

"message": "Internal error, traceback\_id=12345678"

}

}

An internal error response can also have text content, for example when Safe Avenue is under maintenance.

The following are all possible HTTP statuses that Safe Avenue API returns:

| **HTTP status** | **Meaning** |
| --- | --- |
| 200 | Request successfully processed |
| 400 | Client error or parameter validation error |
| 401 | Authentication required or failed |
| 403 | Permission denied, or request is understood but unable to fulfill due to logical error |
| 404 | An object as referenced in parameters is not found |
| 405 | Unrecognized or wrong HTTP method |
| 500 | Internal error, unexpected error |
| 502 | Internal error, possibly server down |
| 503 | Service is temporarily unavailable, possibly due to overload |

Refer to [errors](#errors) for complete list of error codes and corresponding HTTP status.

## Data types

Data types for request parameters and response JSON content, they are written in PascalCase and are used throughout this document.

### Simple types

Simple types:

Integer

Size

Boolean = true | false

String

Flag = 1 | 0

DateTime = String

List = [Type]

EmailAddress

PhoneNumber

Size is positive integer.

Flag is boolean used for procedure parameters, 1 for true, 0 for false.

DateTime is string with format YYYY-MM-DDThh:mm:ssZ representing UTC timestamp. For example, "2013-03-26T14:32:01Z" is 26 Mar 2013 14:32:01 in UTC time zone.

A List is represented with a type in square brackets. For example, [Integer] is a list of integers.

An EmailAddress is a email address in {username}@{domain} format, maximum length is 254 characters. For example: example-email-address@f-secure.com.

PhoneNumber has the format {prefix}{country\_code}{number} with + prefix, 2 digits country code and the rest of the number. Phone number length is from 9 to 20 characters. For example: +60123456789

### Product type

A product:

Product = {

product\_code: String

}

product\_code

Product code identifies a Safe Avenue product, for example "helios\_win", it is configured by F-Secure SIE. To get a list of provisionable product code, see [GET get\_product\_list()](#get-get-product-list).

### Customer type

A Customer represents an end user of Safe Avenue:

Customer = {

extref: String,

oneid\_uuid: String,

customer\_type: CustomerType,

license\_size: Size,

expiry\_date: DateTime,

customer\_identifier: String,

quota: Size,

status: CustomerStatus

}

CustomerType = "CONTINUOUS" | "PREDEFINED"

CustomerStatus = "VALID" | "SUSPENDED" | "EXPIRED"

extref

A customer external reference uniquely identifies Safe Avenue customer, extref is set by operator and is usually the same as the operator's own internal customer id for easy mapping between operator customer and Safe Avenue customer. A valid extref must match the regular expression ^[-a-zA-ZÀ-ÖÙ-ßà-öø-ǽΆ-ώぁ-ゞァ-ヾА-Яа-я一-﨩0-9\_.s@']{1,64}$.

oneid\_uuid

A unique identifier that references a customer's [user account](#user-account)

customer\_type and status

Predefined customers have expiry date and expire if expiry date is over. Continuous customers do not expire. Customer status is valid initially, suspended after [POST suspend\_customer()](#post-suspend-customer) call.

license\_size

The number of licenses can be provisioned for the customer, maximum value allowed for license size is 256.

expiry\_date

Expiry date for predefined customer, a customer is expired if expiry date is less than current date. For example if expiry date is 2015-01-01 and current date is 2015-01-02 then the predefined customer has expired.

quota

[Storage quota](#storage-quota) size in megabyte.

An example of Customer object:

{

extref: 'ASDFZXCV',

license\_size: 7

status: 'VALID',

customer\_identifier: 'ASDF-ZXCV-QWER-UIOP',

customer\_type: 'PREDEFINED',

expiry\_date: '2012-12-12T23:59:59Z',

quota: 10000

}

### Device type type

Device platform:

DeviceType = String

Below is the possible values for DeviceType.

| **DeviceType** | **Platform** |
| --- | --- |
| "mac" | Macintosh |
| "mac\_mega" | Macintosh (**deprecated**) |
| "mob" | Generic mobile platform (**deprecated**) |
| "mob-android" | Android |
| "mob-bb" | BlackBerry (**deprecated**) |
| "mob-ios" | iOS (mobile) |
| "mob-symbian" | Symbian (**deprecated**) |
| "mob-winmo" | Windows Mobile (**deprecated**) |
| "mob-winphone" | Windows Phone |
| "win\_pc" | Windows PC |

### License type

A License represents a customer's subscription to a Product:

License = {

product\_code: String,

device\_type\_code: DeviceType,

device\_name: String,

license\_uuid: String,

installer\_key: String,

installer\_url: String

}

product\_code

Product code of this license.

device\_type\_code

Device platform of the license.

device\_name

Device name of the license. See [POST rename\_device()](#post-rename-device) for valid device name format.

license\_uuid

Unique identifer of the license.

installer\_key

Installer key of the license, which is prompted during product installation.

installer\_url

URL to the license's product installer.

An example of License object:

{

product\_code: "helios\_win",

device\_type\_code: "win\_pc",

device\_name: "Home PC",

license\_uuid: "a95f8148-5221-4359-a62e-bb7d678ba062",

installer\_key: "C-UZEHD-PDNHN-GDXFQ-LE6E9",

installer\_url: "http://example.f-secure.com/installer\_C-UZEHD-PDNHN-GDXFQ-LE6E9\_.exe",

}

### Customer session type

A Safe Avenue Iframe session for a customer:

CustomerSession = {

token: String,

url: String,

browsed: Boolean

}

token

Identifier of the CustomerSession.

url

Safe Avenue Iframe URL relative to server, for example:

"/iframe/-v3/LAST\_DEPLOYMENT/s/helios/chewbo/c7fec0c8ca5649f680f53e94/html?i=0"

browsed

Is the Iframe URL already accessed by the customer?

### Locale type

Language code:

Locale = String

Below are the possible values for Locale.

| **Locale** | **Language** |
| --- | --- |
| "da" | Danish |
| "de" | German |
| "en" | English |
| "es" | Spanish |
| "fi" | Finnish |
| "fr" | French |
| "it" | Italian |
| "ja" | Japanese |
| "nl" | Dutch |
| "no" | Norwegian |
| "pl" | Polish |
| "pt" | Portuguese |
| "pt-br" | Brazilian Portuguese |
| "ru" | Russian |
| "sv" | Swedish |
| "tr" | Turkish |
| "zh-hk" | Traditional Chinese (Hong Kong) |
| "zh-tw" | Traditional Chinese (Taiwan) |
| "vi" | Vietnamese |

## Procedures

A procedure in Safe Avenue API is an HTTP resource that can be called to query or update data.

Each procedure returns either data (success result) or error. [Common errors](#common-errors) are implied and may not be repeated in the procedure's documentation.

Procedure title in sections below has the format {http\_method} {procedure\_name}(), for example POST create\_customer()

http\_method

The only allowed HTTP method for the procedure.

procedure\_name

Name of the procedure.

### GET echo()

Return back the provided message. The purpose of this procedure is to let client tests that API call works.

**Parameters**

message: String

Echo message.

**Return** echo message:

{

message: String

}

### GET get\_product\_list()

Get a list of products that can be used to call [POST provision\_license()](#post-provision-license).

**No parameter**

**Return** list of products:

{

products : [Product]

}

### POST send\_download\_email()

Send an email to customer with instructions and link to product download page.

The download email template can be customized if needed.

Product download page detects user device platform so that the user obtains product installer suitable for his or her device.

**Parameters**

email\_addr: EmailAddress

Email address to send to.

locale: Locale

Locale of the email content.

**Returns** {}

**Errors**

invalid\_locale

If provided locale is invalid.

invalid\_email\_address

If provided email address has invalid format.

### POST send\_download\_sms()

Send an SMS to customer with instructions and link to product download page.

The download email template can be customized if needed.

Product download page detects user device platform so that the user obtains product installer suitable for his or her device.

**Parameters**

phone\_number: PhoneNumber

Phone number to send SMS to.

locale: Locale

Locale of the SMS content.

**Returns** {}

**Errors**

invalid\_locale

If provided locale is invalid.

invalid\_phone\_number

If provided phone number has invalid format.

sms\_too\_long

If the SMS text to send is longer than 160 bytes. This happens if the length of SMS text generated from SMS template and dynamic data (e.g. product download link) exceeds 160 bytes.

sms\_limit\_reached

If more than 3 SMS messages are sent to a phone number in less or equal to 60 seconds.

### POST create\_customer()

Create a customer, and optionally create a [user account](#user-account) then link to the customer.

create\_customer() with user account is actually a shorthand for plain create\_customer() and name\_customer(), for example:

create\_customer(extref, username, email\_addr, first\_name, last\_name)

Is the same as:

create\_customer(extref)

name\_customer(extref, username, email\_addr, first\_name, last\_name)

If any user account parameter is provided, name\_customer() is invoked, so user account parameters validation follows name\_customer()'s.

If create\_customer() with user account failed with internal\_error, client can [retry create\_customer()](#retry-create-customer).

**Parameters**

extref: String

Customer extref.

license\_size: Size

Optional, default 0. License size of the created customer.

customer\_type: CustomerType

Optional, default "CONTINUOUS". Customer type.

expiry\_date: DateTime()

Optional, required only if customer\_type = "PREDEFINED". Expiry date for the predefined customer.

quota\_size: Size

Optional, default 0. Customer [storage quota](#storage-quota) size.

See [POST name\_customer()](#post-name-customer) for more info on the parameters below.

username: String

Optional. Username of the user account.

email\_addr: EmailAddress

Optional. Email address of the user account.

first\_name: String

Optional. First name of the user account.

last\_name: String

Optional. Last name of the user account.

user\_locale: Locale

Optional. Locale of the user account.

send\_email: Flag

Optional, default 1. Send welcome email after creating user account?

**Returns** the created customer:

{

customer: Customer

}

**Errors**

invalid\_customer\_extref

If the specified extref is invalid, see [customer type](#customer-type).

already\_exists

If the specified extref already in use.

license\_limit\_reached

If the specified license\_size is greater than [maximum license size](#maximum-license-size).

expiry\_date\_already\_passed

If the specified expiry\_date has already passed.

expiry\_date\_invalid

If the given expiry\_date is greater than 2030-12-20.

invalid\_storage\_quota

If the specified quota is invalid, see [storage quota](#storage-quota).

storage\_not\_available

If storage is not configured and quota is specified.

username\_already\_used

See [retry create\_customer()](#retry-create-customer).

email\_address\_already\_used

See [retry create\_customer()](#retry-create-customer).

error\_creating\_customer

See [retry create\_customer()](#retry-create-customer).

param\_mismatch

See [retry create\_customer()](#retry-create-customer).

extref\_mismatch

See [retry create\_customer()](#retry-create-customer).

[POST name\_customer()](#post-name-customer) errors also apply here if user account parameters are provided.

### POST name\_customer()

Name a customer by creating a [user account](#user-account) and link with the customer.

Storage will be provisioned if conditions are met, see [storage quota](#storage-quota).

When using FSIDP, username is always the same as email address. Also, for first name and last name:

* Length must be less or equal to 256 characters.
* Must not contain characters \ / : ? \* < > " | ( ) ;

When using SSO, username need not be same as email adddress, first name and last name validation depends on operator's identity provider.

**Parameters**

extref: String

Customer extref.

username: String

Optional and default to email\_addr if using FSIDP, otherwise not set.

email\_addr: EmailAddress

Email address of the user account.

first\_name: String

First name of the user account.

last\_name: String

Last name of the user account.

user\_locale: Locale

Optional, default to configured operator locale. Locale of the user account.

send\_email: Flag

Always 1 if using FSIDP, otherwise optional and default to 1. Send welcome email after creating user account?

**Returns** the named customer:

{

customer: Customer

}

**Errors**

invalid\_locale

If user\_locale is invalid.

does\_not\_exist

If the customer does not exist.

invalid\_user\_details

If user details (email address, first name, last name) are invalid.

naming\_constraint\_error

If any one of the following criteria is met:

* When using FSIDP but username is not the same as email address.
* When using FSIDP but send\_email = 0.
* When using SSO but username is not provided.

customer\_already\_named

If the customer is already named.

user\_already\_exists

The user with same username already exists.

group\_user\_already\_exists

The user with same username already exists in operator group.

customer\_locked

See [customer\_locked error](#customer-locked-error).

### GET get\_customer()

Get details of a customer.

**Parameters**

extref: String

Customer extref.

include\_licenses: Boolean

Optional, default 0. Include licenses data in response?

**Returns** the queried customer and active licenses if include\_licenses = 1:

{

customer: Customer,

licenses: [License]

}

**Errors**

does\_not\_exist

If the customer does not exist.

### POST rename\_customer\_extref()

Rename a customer extref.

When using SSO, extref of a customer that is already linked to a user account cannot be renamed.

**Parameters**

extref: String

Existing customer extref.

new\_extref: String

New customer extref.

**Returns** old and new extrefs:

{

old\_extref: String,

new\_extref: String

}

**Errors**

does\_not\_exist

If the customer does not exist.

invalid\_customer\_extref

If the customer extref is invalid, see [customer type](#customer-type).

already\_exists

If new\_extref already in use.

naming\_constraint\_error

When using SSO and the customer already linked to a user account.

customer\_locked

See [customer\_locked error](#customer-locked-error).

illegal\_state\_error

See [illegal\_state error](#illegal-state-error).

### POST update\_customer()

Update a customer:

* Convert customer type from continuous to predefined or vice versa
* Update license size
* Update predefined customer expiry date
* Provision storage

**Parameters**

extref: String

Customer extref.

license\_size: Size

Optional. New license size.

customer\_type: CustomerType

Optional. New customer type.

expiry\_date: DateTime

Optional. New expiry date.

quota\_size: Size

Optional. New storage quota.

force\_downgrade: Flag

Optional, default 0. Force downgrade means to terminate oldest licenses when new license size set is less than number of active licenses to prevent license\_limit\_reached error from being raised. For example you have a customer with license size 3 and provisioned 3 licenses:

api.create\_customer(extref='cust001', license\_size=3)

api.provision\_license(extref='cust001', product\_code='product1')

api.provision\_license(extref='cust001', product\_code='product2')

api.provision\_license(extref='cust001', product\_code='product3')

Trying to update license size to 2 will fail because there are 3 active licenses:

api.update\_customer(extref='cust001', license\_size=2)

// license\_limit\_reached error

But if force\_downgrade = 1:

api.update\_customer(extref='cust001', license\_size=2, force\_downgrade=1)

// product1 license terminated, leaving product2 and product3 licenses

The customer license size will be successfully set to 2 and oldest licenses (here only product1 license) are terminated to make sure number of active licenses is not more than license size.

**Returns** the updated customer:

{

customer: Customer

}

**Errors**

does\_not\_exist

If the customer does not exist

license\_limit\_reached

If the specified license\_size is less than existing license size and force\_downgrade = 1, or the specified license\_size is greater than [maximum license size](#maximum-license-size).

expiry\_date\_already\_passed

If the specified expiry\_date has already passed.

invalid\_storage\_quota

If quota is invalid, see [storage quota](#storage-quota).

storage\_not\_available

If storage is not configured and quota is specified.

customer\_locked

See [customer\_locked error](#customer-locked-error).

incomplete\_update

See [incomplete\_update error](#incomplete-update-error).

illegal\_state\_error

See [illegal\_state error](#illegal-state-error).

### POST suspend\_customer()

Suspend a customer.

See also [suspended customer](#suspended-customer).

**Parameters**

extref: String

The customer extref.

**Returns** {}

**Errors**

does\_not\_exist

If the customer does not exist.

customer\_locked

See [customer\_locked error](#customer-locked-error).

incomplete\_update

See [incomplete\_update error](#incomplete-update-error).

illegal\_state\_error

See [illegal\_state error](#illegal-state-error).

### POST resume\_customer

Resume a suspended customer.

See also [customer statuses](#customer-statuses).

**Parameters**

extref: String

The customer extref.

**Returns** {}

**Errors**

does\_not\_exist

If the customer does not exist.

customer\_locked

See [customer\_locked error](#customer-locked-error).

incomplete\_update

See [incomplete\_update error](#incomplete-update-error).

illegal\_state\_error

See [illegal\_state error](#illegal-state-error).

### POST get\_customer\_session()

Get an existing and updated customer session, or create a new one. This procedure will return a new customer session if token is not specified, is invalid or the customer session already expired, otherwise the customer session will be updated.

See also [customer session](#customer-session).

**Parameters**

extref: String

The customer extref.

token: String

Optional. Token of existing customer session to update.

locale: Locale

Optional. Locale of the customer session. If not provided the customer session locale will be based on user browser preference.

**Returns** the customer session:

{

customer\_session: CustomerSession

}

**Errors**

does\_not\_exist

If the customer does not exist.

invalid\_locale

If provided locale is invalid.

### POST delete\_customer\_session()

Delete a customer session to invalidate the Iframe session.

See also [customer session](#customer-session).

**Parameters**

extref: String

The customer extref.

token: String

The customer session token.

**Returns** {}

**Errors**

does\_not\_exist

If the customer or customer session does not exist.

### POST provision\_license()

Provision a license for a customer.

If the customer type is predefined then the provisioned license have the same expiry date as the customer's.

**Parameters**

extref: String

The customer extref.

product\_code: String

Product code of the license to provision. See also [GET get\_product\_list()](#get-get-product-list).

device\_name: String

Optional, default "My Device {n}" where n is incremented number. Device name of the provisioned license. See POST rename\_device() for more info on device name.

**Returns** the provisioned license:

{

license: License

}

**Errors**

does\_not\_exist

If the customer or product does not exist.

invalid\_device\_name

If the specified device\_name is invalid, see [POST rename\_device()](#post-rename-device).

customer\_not\_named

If the customer is not linked with a user account but this license requires it.

license\_limit\_reached

If remaining license size is 0.

customer\_locked

See [customer\_locked error](#customer-locked-error).

illegal\_state\_error

See [illegal\_state error](#illegal-state-error).

### POST terminate\_license()

Terminate a customer's license.

When a license is terminated, the customer will see in his device that the product subscription has expired and must renew the subscription to continue using it.

A terminated license is not an active license, so it will not be returned from [GET get\_customer()](#get-get-customer).

A terminated license can become active again by calling [POST reactivate\_license()](#post-reactivate-license).

**Parameters**

extref: String

The customer extref.

license\_uuid: String

The license UUID.

**Returns** {}

**Errors**

does\_not\_exist

If the customer or license does not exist.

customer\_locked

See [customer\_locked error](#customer-locked-error).

illegal\_state\_error

See [illegal\_state error](#illegal-state-error).

### POST reactivate\_license()

Reactivate a terminated license of a customer.

**Parameters**

extref: String

The customer extref.

license\_uuid: String

The license UUID.

**Returns** {}

**Errors**

does\_not\_exist

If the customer or license does not exist.

license\_limit\_reached

If remaining license size is 0.

product\_inactive

If the product of the license is inactive. A product is marked as inactive when the product is intended to be removed.

customer\_locked

See [customer\_locked error](#customer-locked-error).

illegal\_state\_error

See [illegal\_state error](#illegal-state-error).

### POST rename\_device()

Update device name of a license.

Device name has the length from 1 to 30 characters, first and last characters must be alphanumeric.

**Parameters**

extref: String

The customer extref.

license\_uuid: String

The license UUID.

device\_name: String

New device name.

**Returns** {}

**Errors**

does\_not\_exist

If the customer or license does not exist.

invalid\_device\_name

If the specified device\_name is invalid.

## Errors

| **HTTP status** | **Error code** |
| --- | --- |
| 400 | expiry\_date\_already\_passed |
| 400 | expiry\_date\_invalid |
| 400 | extref\_mismatch |
| 400 | invalid\_customer\_extref |
| 400 | invalid\_device\_name |
| 400 | invalid\_email\_address |
| 400 | invalid\_locale |
| 400 | invalid\_phone\_number |
| 400 | invalid\_storage\_quota |
| 400 | invalid\_user\_details |
| 400 | param\_mismatch |
| 400 | parameters\_not\_match |
| 401 | unauthorized |
| 403 | already\_exists |
| 403 | customer\_already\_named |
| 403 | customer\_locked |
| 403 | customer\_not\_named |
| 403 | email\_address\_already\_used |
| 403 | error\_creating\_customer |
| 403 | group\_user\_already\_exists |
| 403 | illegal\_state |
| 403 | license\_limit\_reached |
| 403 | naming\_constraint\_error |
| 403 | permission\_denied |
| 403 | product\_inactive |
| 403 | sms\_limit\_reached |
| 403 | sms\_too\_long |
| 403 | storage\_not\_available |
| 403 | user\_already\_exists |
| 403 | username\_already\_used |
| 404 | does\_not\_exist |
| 405 | method\_not\_allowed |
| 500 | incomplete\_update |
| 500 | internal\_error |
| 503 | server\_busy |

Common errors that can be returned by all procedures:

internal\_error

Unexpected server error

permission\_denied

Current API user does not have permission to call the procedure

method\_not\_allowed

Unrecognized or wrong HTTP method

parameters\_not\_match

Invalid parameter or invalid combination of parameters

server\_busy

The procedure cannot handle the request due to limited resources

### customer\_locked error

To prevent customer data from being corrupted due to concurrently update, a customer is locked before being update and unlocked after update. If another update tries to lock the customer before the first update finishes, then customer\_locked error will occur.

### incomplete\_update error

Due to some limitations of Safe Avenue, certain operations are not atomic. For example, suspend\_customer() might return incomplete\_update error, leaving the customer in inconsistent state. Client should repeat the procedure when encountering incomplete\_update error until success, in this case call suspend\_customer() again. If the error happens consistently then it is probably a bug, the only way to resolve this is to contact F-Secure.

## Customer session

A customer session is an Iframe sesssion, each customer can have at most 10 customer sessions. POST get\_customer\_session() can be used to create and update a customer session, POST delete\_customer\_session() to invalidate a customer session. A customer session length is 2 hours.

For example, to create new iframe:

cust\_session = api.get\_customer\_session(extref='cust001')

render\_iframe(cust\_session.url) # Use session.url as iframe src

http\_session.cust\_session = cust\_session

Then to re-render the iframe and extend the customer session length:

cust\_session = http\_session.cust\_session

cust\_session = api.get\_customer\_session(extref='cust001', token=cust\_session.token)

render\_iframe(cust\_session.url)

http\_session.cust\_session = cust\_session

Or to re-render the iframe without extending the customer session length:

cust\_session = http\_session.cust\_session

render\_iframe(cust\_session.url)

To invalidate the customer session:

cust\_session = http\_session.cust\_session

api.delete\_customer\_session(extref='cust001', token=cust\_session.token)

http\_session.cust\_session = null

An invalidated customer session is effectively a non-existing customer session, using invalidated customer session and non-existing customer session will result in session expired page.

Token is the customer session identifier, by providing valid token to get\_customer\_session no new customer session is created. This way there can be 10 web sites with each different customer session, deleting customer session from one site will not cause other sites' customer session to invalidate.

## Customer statuses

A customer can be either valid, expired or suspended.

A customer is valid when customer is in normal state, this is the state of the customer initially.

A predefined customer is expired when its expiry date has passed, continuous customer cannot be expired.

A customer is suspended when [POST suspend\_customer()](#post-suspend-customer) is called, the customer will then be valid after POST resume\_customer().

When a customer is suspended:

* The licenses are also suspended, a suspended license is like [terminated license](#terminated-license), the customer won't be able to use it.
* If storage quota is provisioned, it will be suspended. The customer can still download his storage content for another 30 days, after that the storage will be purged.

illegal\_state error is raised when an illegal operation is invoked on wrong customer status. For example [POST provision\_license()](#post-provision-license) on suspended customer.

## User account

A user account allows Safe Avenue to authenticate a user.

A customer may or may not be linked with a user account. For example:

create\_customer(extref='cust001')

# Customer 'cust001' is not linked with user account

name\_customer(extref='cust001',

email\_addr='cust001@mailinator.com',

first\_name='Cust',

last\_name='001')

# Now the customer is linked with a user account

Customer can also be created and linked with user account in one call:

create\_customer(extref='cust001',

email\_addr='cust001@mailinator.com',

first\_name='Cust',

last\_name='001')

Creating customer and link new user account in this way is not atomic, the user account might be created but linking might fail. To prevent user\_already\_exists error on second call using same user account, POST create\_customer() accepts existing user account (username and email address). For example:

# Assume this one fail

create\_customer(extref='cust001',

email\_addr='cust001@mailinator.com',

first\_name='Customer',

last\_name='One')

# Client can retry again but username or email\_addr must be the same

create\_customer(extref='cust001',

email\_addr='cust001@mailinator.com',

first\_name='Customer',

last\_name='One')

Note that when retrying create\_customer() username and emaill address must be the same, otherwise param\_mismatch will be raised. If the user account cannot be found, error\_creating\_customer will be raised. If the user account cannot be associated with the customer extref, extref\_mismatch will be raised. If the user account already linked with another customer, username\_already\_used (SSO) or email\_address\_already\_used (FSIDP) will be raised.

## Storage quota

Customer storage quota is the number of megabyte (not mebibyte) that the customer has in cloud storage. Customer storage can be provisioned from [POST create\_customer()](#post-create-customer) and [POST update\_customer()](#post-update-customer). When the following conditions are met, then storage is provisioned:

* Quota is greater than 0
* Quota is equal to trial quota or greater than maximum app store quota
* Customer is linked with a user account

Trial quota and maximum app store quota are configurable per operator.