SAFE 3 INTEGRATION MANUAL  
FOR SSO

Table of Contents

[Introduction 6](#_Toc435098702)

[User Experience integration - SAFE components 7](#_Toc435098703)

[SAFE Home 7](#_Toc435098704)

[Accessing SAFE Home 7](#_Toc435098705)

[SAFE Clients 8](#_Toc435098706)

[Activation user journey 9](#_Toc435098707)

[SAFE Client Activation sequence diagram 10](#_Toc435098708)

[SAFE Client Activation business logic 11](#_Toc435098709)

[SAFE Client Activation content sources 12](#_Toc435098710)

[SAFE emails 14](#_Toc435098711)

[Co-Branding 15](#_Toc435098712)

[Welcome email for a customer registering through the Operator 16](#_Toc435098713)

[Installation email 17](#_Toc435098714)

[SAFE SMS's 18](#_Toc435098715)

[F-Secure sending SMSs 18](#_Toc435098716)

[SMSc Integration solutions 18](#_Toc435098717)

[User Experience integration – Use cases 20](#_Toc435098718)

[PROVISIONING use cases 20](#_Toc435098719)

[UCP-1: A Customer purchases SAFE from the Operator 20](#_Toc435098720)

[User experience example: 20](#_Toc435098721)

[Sequence diagram: 21](#_Toc435098722)

[UCP-2: A Customer registers for a trial after downloading a SAFE Client application 23](#_Toc435098723)

[UCP-3 A Customer upgrades a trial subscription to a paid via the Operator 26](#_Toc435098724)

[UCP-3b: A Customer buys more licenses via in-app purchase 28](#_Toc435098725)

[UCP-4 A Customer upgrades the license size of a purchased subscription 29](#_Toc435098726)

[UCP-5 A Customer downgrades the license size of a purchased subscription 31](#_Toc435098727)

[Client installation use cases 37](#_Toc435098728)

[UCI-1: User triggers SAFE Client installation from SAFE Home to the device that they are currently using 38](#_Toc435098729)

[UCI-2: User triggers SAFE Client installation from an installation email 40](#_Toc435098730)

[UCI-3: User triggers SAFE Client installation from an installation SMS 42](#_Toc435098731)

[UCI-4: A Customer tries to install a SAFE Client, but all licenses are in use. The Customer frees a license to proceed 44](#_Toc435098732)

[Family use cases 45](#_Toc435098733)

[UCF-1: A Customer invites a family member to share the service 45](#_Toc435098734)

[Deployment of the client applications 46](#_Toc435098735)

[Mobile clients in AppStores 46](#_Toc435098736)

[API integration 47](#_Toc435098737)

[Versioning and compatibility 47](#_Toc435098738)

[URL and authentication 47](#_Toc435098739)

[GET vs POST 48](#_Toc435098740)

[Request Headers 48](#_Toc435098741)

[Content-Type 48](#_Toc435098742)

[X-Transaction 48](#_Toc435098743)

[Response 49](#_Toc435098744)

[Data types 50](#_Toc435098745)

[Simple types 50](#_Toc435098746)

[Product type 51](#_Toc435098747)

[Product Category type 51](#_Toc435098748)

[Customer type 52](#_Toc435098749)

[Subuser type 54](#_Toc435098750)

[Device type 54](#_Toc435098751)

[License type 55](#_Toc435098752)

[Customer session type 56](#_Toc435098753)

[User info type 56](#_Toc435098754)

[Locale type 57](#_Toc435098755)

[Procedures 58](#_Toc435098756)

[GET echo() 58](#_Toc435098757)

[GET get\_product\_list() 58](#_Toc435098758)

[POST send\_welcome\_email() 59](#_Toc435098759)

[POST send\_welcome\_sms() 59](#_Toc435098760)

[POST send\_download\_email() 60](#_Toc435098761)

[POST send\_download\_sms() 60](#_Toc435098762)

[POST create\_customer() 61](#_Toc435098763)

[POST name\_customer() 64](#_Toc435098764)

[POST create\_user() 66](#_Toc435098765)

[POST delete\_user() 68](#_Toc435098766)

[GET get\_user() 68](#_Toc435098767)

[POST add\_user\_to\_customer() 69](#_Toc435098768)

[POST remove\_user\_from\_customer() 70](#_Toc435098769)

[POST update\_user\_quota\_blocks() 70](#_Toc435098770)

[GET get\_customer() 71](#_Toc435098771)

[POST rename\_customer\_extref() 72](#_Toc435098772)

[POST update\_user() 73](#_Toc435098773)

[POST update\_customer() 74](#_Toc435098774)

[POST suspend\_customer() 77](#_Toc435098775)

[POST resume\_customer 77](#_Toc435098776)

[POST get\_customer\_session() 78](#_Toc435098777)

[POST delete\_customer\_session() 79](#_Toc435098778)

[POST provision\_license() 79](#_Toc435098779)

[POST terminate\_license() 80](#_Toc435098780)

[POST rename\_device() 81](#_Toc435098781)

[Errors 81](#_Toc435098782)

[Common errors 83](#_Toc435098783)

[Internal error 83](#_Toc435098784)

[Permission denied error 83](#_Toc435098785)

[Method not allowed error 83](#_Toc435098786)

[Parameters not match error 83](#_Toc435098787)

[Server busy error 83](#_Toc435098788)

[Customer locked error 83](#_Toc435098789)

[Incomplete update error 83](#_Toc435098790)

[Incomplete group update error 84](#_Toc435098791)

[Customer session 84](#_Toc435098792)

[Customer statuses 85](#_Toc435098793)

[Suspended customer 85](#_Toc435098794)

[Illegal state 85](#_Toc435098795)

[User account 85](#_Toc435098796)

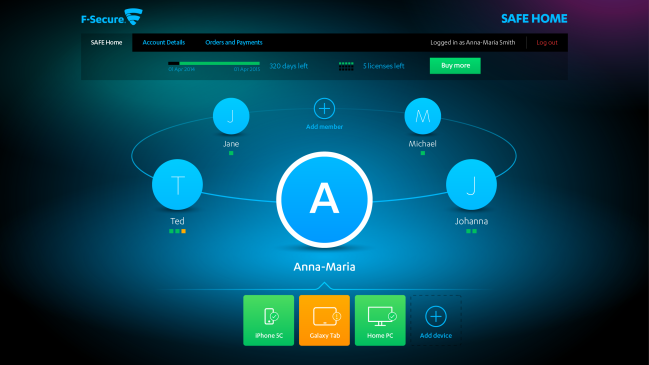
[Storage quota 87](#_Toc435098797)

# Introduction

Slightly more complex to integrate, the "Operator Account" integration variant can potentially provide a stronger user experience. For this model, a browser-based SSO integration is built between the F-Secure OneID platform and the Operator Identity Provider. Users will see the familiar operator login screens within SAFE Components.  
This scenario does require a separate SSO integration project between the Operator and F-Secure, and does also mean additional implementation to make the "Family" model of grouped users possible: as F-Secure cannot add accounts to the Operator Identity Provider database, an additional integration element is required to handle the scenario of customers inviting other users to share their license quota.  
For an integration to F-Secure SAFE to be considered successful, the service has to be easily discoverable and usable by the end user. To this end, F-Secure has done extensive studying of the user journeys, and this document is broken into user journeys that must be implemented by the Operator for their integration to SAFE.

# User Experience integration - SAFE components

## SAFE Home

  
SAFE Home (the evolution of the F-Secure Avenue iFrame) is the control panel at the heart of SAFE. Using SAFE Home, users are able to:

Manage protected devices (send installation links by SMS and email, free tokens from devices no longer in use)

Manage the Circle (invite or remove members)

SAFE Home is a web application, thus accessible with a normal web browser.

## Accessing SAFE Home

Buttons to access SAFE Home are built into all SAFE Clients. In addition, SAFE Home must be made accessible from the Operator webpages for registered users. Also, all e-mail sent from SAFE must include a link to SAFE Home, to ensure that users get used to the service.

## SAFE Clients

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

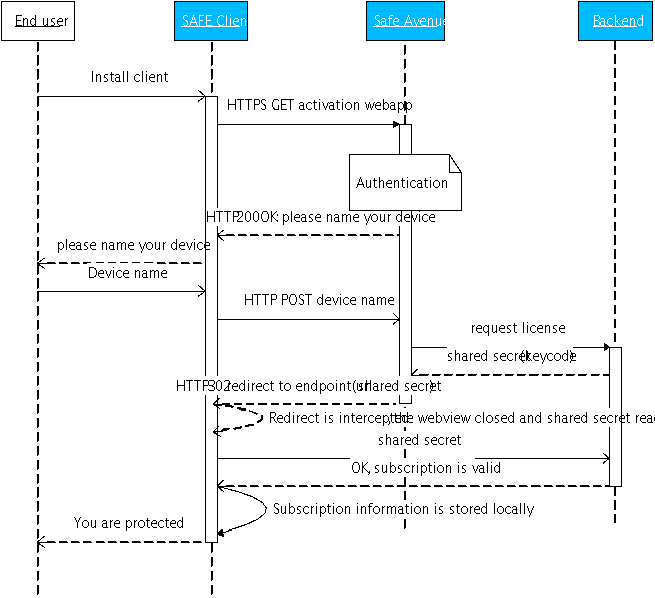
The SAFE Clients are the applications that provide the actual protection of the end users devices. Due to differences between operating systems, the exact functionality provided differs slightly per platform. Regardless, the user experience of the SAFE Clients is designed to be as coherent as possible.

### Activation user journey

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

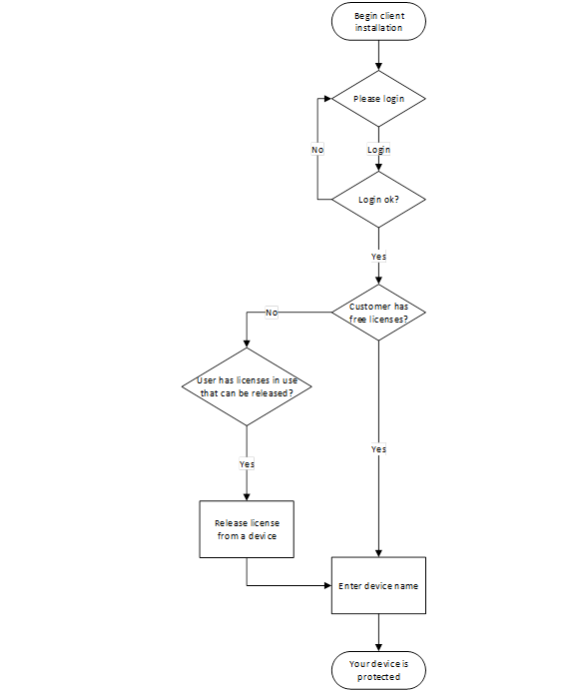

#### 

### ****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, F-Secure 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.

### ****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.

##### 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. In this integration model, the login screen content is served by the Operator’s web servers. |

##### SAFE Business logic

|  |  |
| --- | --- |
| C:\9a35e9daa6c1e471386fdf4b70c3b514 | After successful login, the SAFE business logic is performed as detailed in the previous chapter. |

##### 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.)  


## SAFE emails

  
  
  
As of today, email is the primary communication channel used by SAFE. This is especially true in the "Operator Security Account" integration scenario, where F-Secure provides the IdP, as F-Secure OneID uses an e-mail address as the login name.  
There are currently two subcategories of emails being sent by SAFE:

**Lifecycle messages:** includes all of the emails sent by the F-Secure Avenue platform. These are messages related to the lifecycle of the Customer, such as SAFE Client installation links.

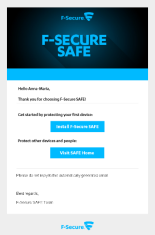
**Identity Provider messages:** emails for which the sender varies depending on the integration method. An example is a "reset password" email, which in the Operator Account integration is being sent by the Operator, and in the Operator Security Account integration is being sent by the F-Secure OneID platform.

### Co-Branding

The exact co-branding possibilities vary slightly based on the platform used to send the emails. The F-Secure Avenue platform allows the Operator to provide two templates (HTML & plain text) for each email. Both are being sent, and the capabilities of the email client of the receiver determine which is being shown. For each different email, a list of tokens are available, which are then replaced by dynamic data. As an example, in an installation email, the token %(installer\_url)s would be replaced by the installation url for the Operator.   
As a lot of freedom is provided to customize the HTML content of the emails, care must be taken with the testing. There are dozens of email clients with relevant market shares, and the level of HTML support provided varies greatly.

* Background images should be avoided due to poor support
* Tables are often the best way to define layouts
* Inline CSS styles often offer better results
* Test thoroughly. There are several services available that make this easier.

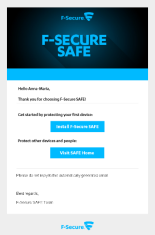
### Welcome email for a customer registering through the Operator

   
In the Operator Account integration, the welcome email is perhaps the single most important user facing component of the service. When the operator provisions a Customer using create\_customer(), sending of this email to the specified email address is triggered. This email includes all the information the user needs to get started:

* Information that a new service is now usable with the Operator account
* Information on what the service includes (license size, potential expiry date)
* Links to:
* SAFE Home
* Client installation

The content of this email is fully customizable by the Operator (HTML & Plain text content provided), but the information listed above must be included. In the Operator account integration variant, the Operator can choose to not send this email via F-Secure components, and instead communicate the welcome message to the end user directly.

### ****Installation email****



The installation email can be triggered in two ways: either the user clicks a button in SAFE Home to protect a device, enters an email address and hits send, or the Operator calls an API method and gives the recipient's email address as parameter. This email includes the following:

* Call to action to protect the device (link)
* Secondary call to action to visit SAFE Home (link) to protect other devices and manage the service

The content of this email is fully customizable by the Operator (HTML & plain text content provided), but the information listed above must be included.

## SAFE SMS's

SMSs have requirement to reach several operators in each country as several licenses per user may have different operator SIMs. This requirement means that either the SMS integration needs to allow sending SMS to several operators and not only partner customers. Therefore the recommendation is to choose first the F-Secure provided 3rd party integrator and depending on estimated traffic and do direct integrations on need basis. F-Secure provided 3rd party integrator has wide support for SMS roaming agreements.

SMSs that are sent by F-Secure have the limitation of the 160 characters. The sender name may be adjusted with some limitations e.g. Partner name may not be used directly unless using dedicated partner integration and limitations on length and no special characters in the sender name. Default sender name in FSecure.

The link in the SMS points to a device detection page that allows the user to download the correct client for the platform of the device that they are using.

SMSC integration or F-Secure sending

As the content of an SMS is limited by length, the most important item to consider regarding SMS communication is the implementation of the sending. The options available are:

**SMSc integration** where F-Secure integrates to the Operator's SMSC. The most obvious benefit of this approach is the ability of the operator to customize the number from which the SMS is sent.

**F-Secure sending,** where F-Secure uses the most economical approach available for SMS delivery. In this scenario, the number from which the SMS are received are usually not customizable.

SMS gateway option is selected based on estimated traffic and below is the list of options.

### F-Secure sending SMSs

Options of SMS gateways that allow sending of SMS without implementation.

F-Secure sending option has the benefits of faster SMS sending and covers very wide support for operators globally and no maintenance allocation required.

1. Nexmo SMS 3rd party global integrator is recommended

F-Secure pays all the SMS and those are then covered in commercial agreement with partner.

### SMSc Integration solutions

SMS options that require implementation or configuration on either partner and/or F-Secure side are listed here. There is also maintenance allocation with these options, costs available by F-Secure sale.

Option C) is HTTPS interface available from F-Secure for partners to integrate. This requires partner to implemented request to send SMSs via F-Secure gateway.

Options B) and D) have integration effort on both sides.

1. F-Secure Generic SMS integration via HTTPS protocol. See the help docs for the API specification and configuration options, provided if option is selected.
2. Configuration via VPN and utilizing standard SMSC protocols e.g. SMPP.
3. Partner specific custom SMSC integration, which is then estimated by F-Secure backend team for workload and schedule.

# User Experience integration – Use cases

## PROVISIONING use cases

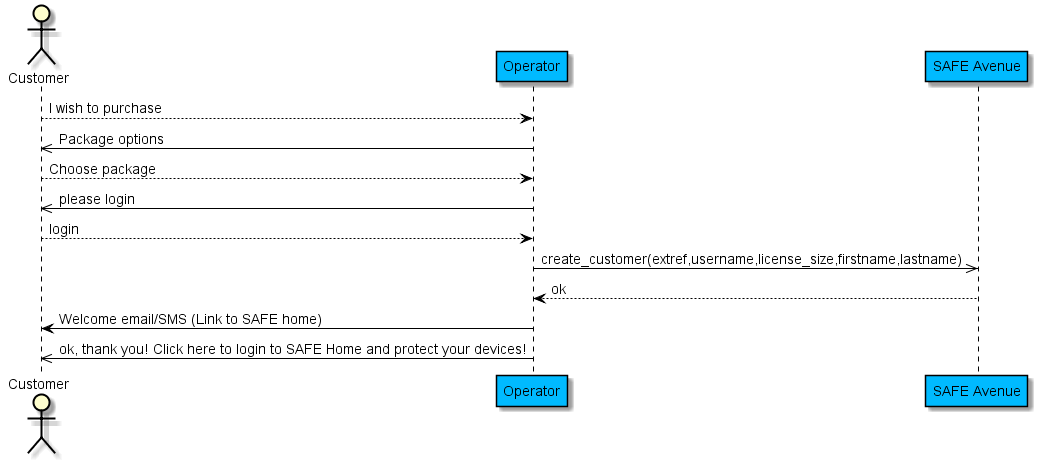
### UCP-1: A Customer purchases SAFE from the Operator

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 received the Welcome email or SMS and is able to start using the service.

### ****User experience example:****

|  |
| --- |
|  |

### ****Sequence diagram:****



**Involved components:**

* Operator sales channels
* F-Secure Avenue API
* SAFE Welcome email

**Integration methods:**

Create\_customer()

For this scenario, the mandatory parameters are:

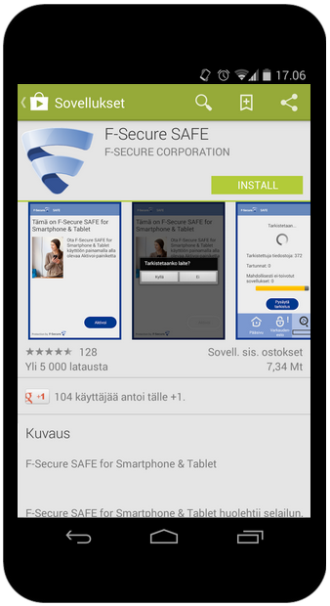
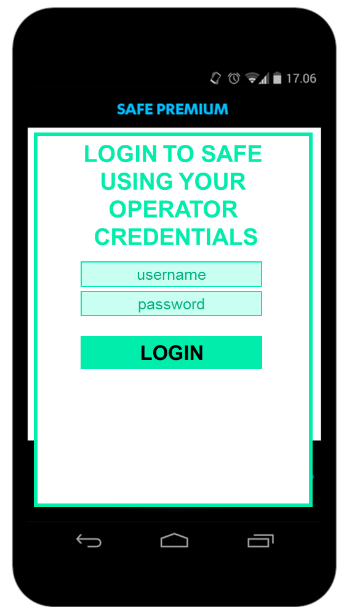
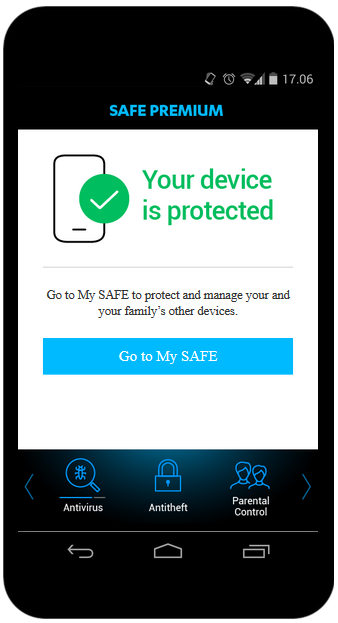
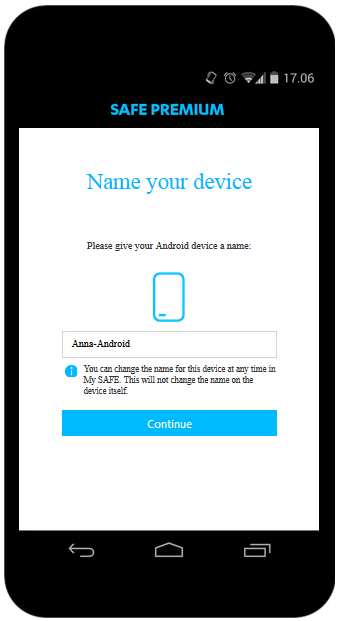
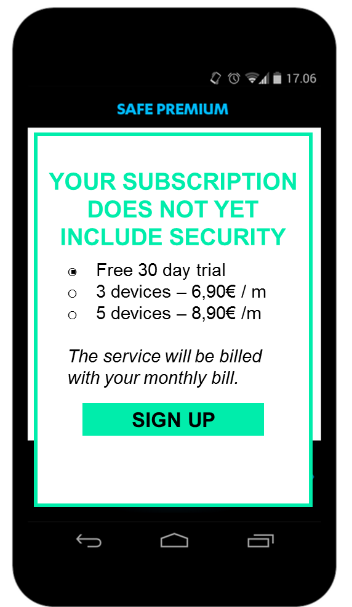
* Extref: reference for the customer in the operator CRM. This reference is used in the future for all API methods to modify the Customer entity in F-Secure Avenue.
* License\_size: the number of devices the Customer is allowed to protect using SAFE.
* Username: reference used in the SSO authentication sequence
* 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:

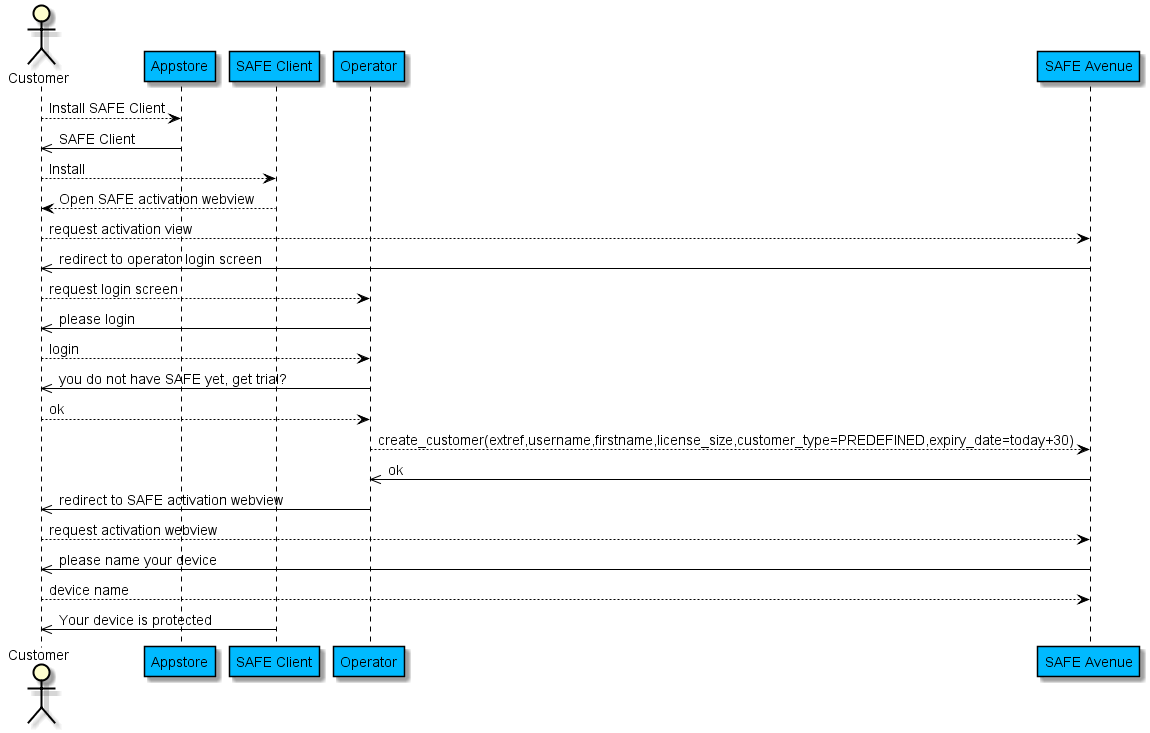
* Locale: if the locale is given, the welcome email will be sent using the template configured for the given locale. If omitted, the welcome email will be sent using the template configured for the default locale of the Operator

### UCP-2: A Customer registers for a trial after downloading a SAFE Client application

SYNOPSIS: A customer finds a SAFE Client application and registers for a trial account during installation.   
EXPECTED END RESULT: The Customer has created an account, protected the first device, and received a confirmation email.   
**User experience example:**

****

**Sequence diagram:**



**Involved components:**

* SAFE Clients

**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 F-Secure Avenue.
* License\_size: the number of devices the Customer is allowed to protect using SAFE.
* Username: reference used in the SSO authentication sequence
* 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
* Customer\_type & expiry\_date: defaults to CONTINUOUS, if creating a predefined trial, should be set to PREDEFINED. If customer\_type is set to PREDEFINED, expiry\_date is required.

If OneID has details of this account already, user can login directly from client. F-Secure Avenue will create trial subscription based on operator configuration (e.g. 3 licenses, 30 days).

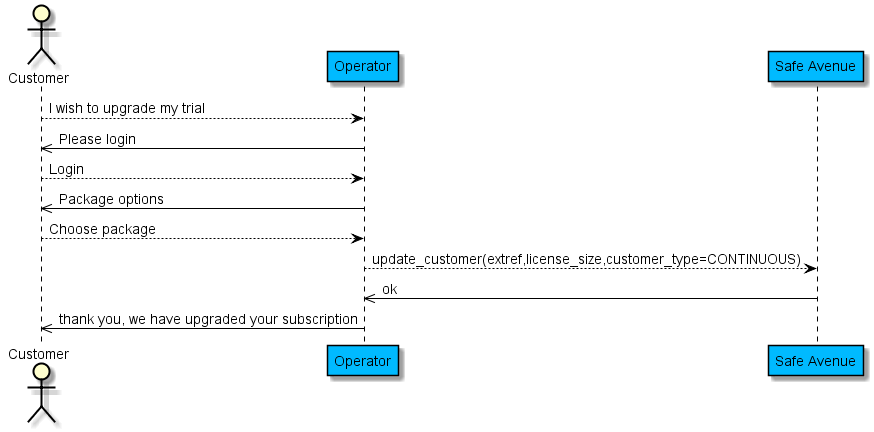
### UCP-3 A Customer upgrades a trial subscription to a paid via the Operator

SYNOPSIS: A customer who has a trial subscription upgrades it to a paid subscription via the Operator.   
EXPECTED END RESULT: The customer's subscription is upgraded to paid.

**User experience example:**

|  |
| --- |
|  |

**Sequence diagram:**



**Involved components:**

* Operator sales channels
* F-Secure Avenue API

**Integration methods:**

Update\_customer()

For this scenario, the mandatory parameters are:

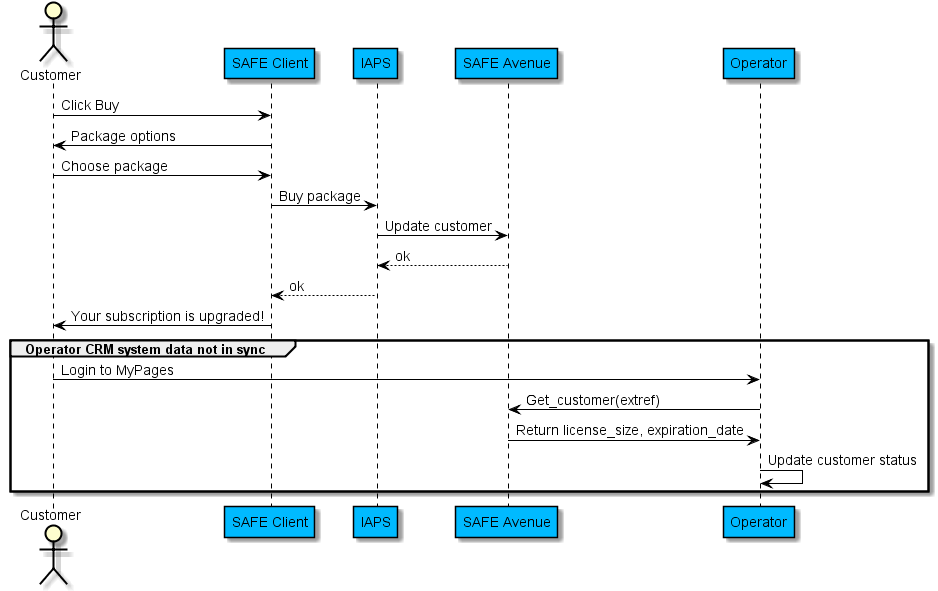
* Extref: the external reference of the customer
* License\_size: the license size matching with the service package purchased by the Customer
* Customer\_type: as the previous subscription was a time-limited trial, the Operator must change it to CONTINUOUS

### UCP-3b: A Customer buys more licenses via in-app purchase

SYNOPSIS: A customer wants to buy more service time or licenses through client via in-app purchase. EXPECTED END RESULT: The purchase is successful. Operator CRM fetches the latest status of the customer via get\_customer() call.

As application stores set strict rules for subscription based products, F-Secure offers in-app purchase capabilities. In such case, the purchase happens outside the operator CRM system, thus operator should initiate get\_customer() call in login to get latest information about the subscription status.

**Sequence diagram:**



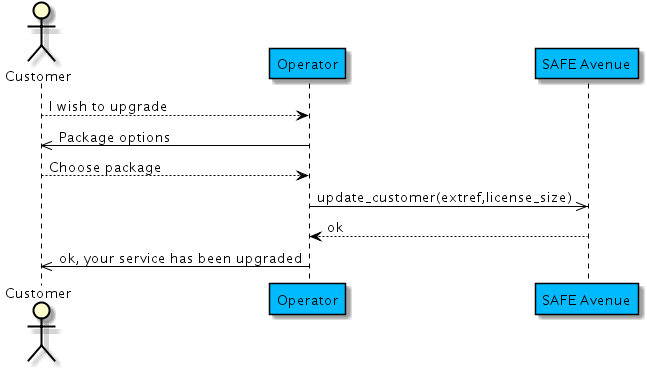
**Involved components:**

* Application stores (AppStore, Google Play)
* Operator CRM
* F-Secure Avenue API
* IAPS (F-Secure In-app purchase service)

### 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
* F-Secure Avenue API

**Integration methods:**

Update\_customer()

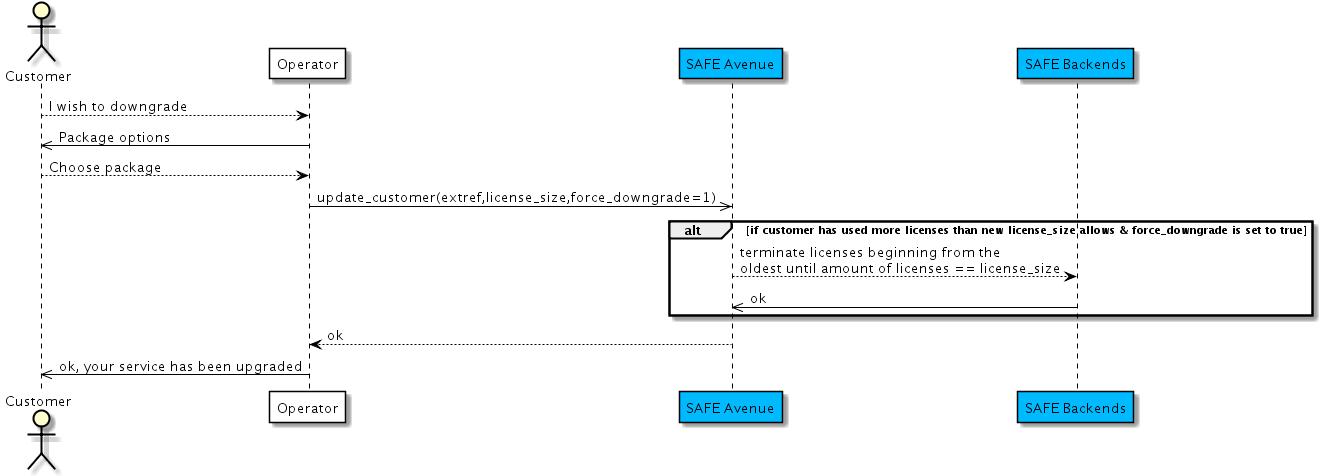
For this scenario, the mandatory parameters are:

* 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
* F-Secure 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

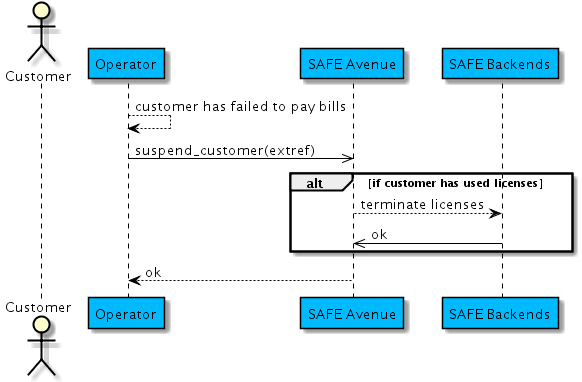
Optional parameters:

* Force\_downgrade: if set to true (1), F-Secure Avenue will force the downgrade of the license\_size. If the customer had used more licenses than the new license\_size allows, F-Secure Avenue will terminate enough licenses to match the new license\_size, beginning from the oldest. If force\_downgrade is not given, F-Secure 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
* F-Secure Avenue API

**Integration methods:**

suspend\_customer()

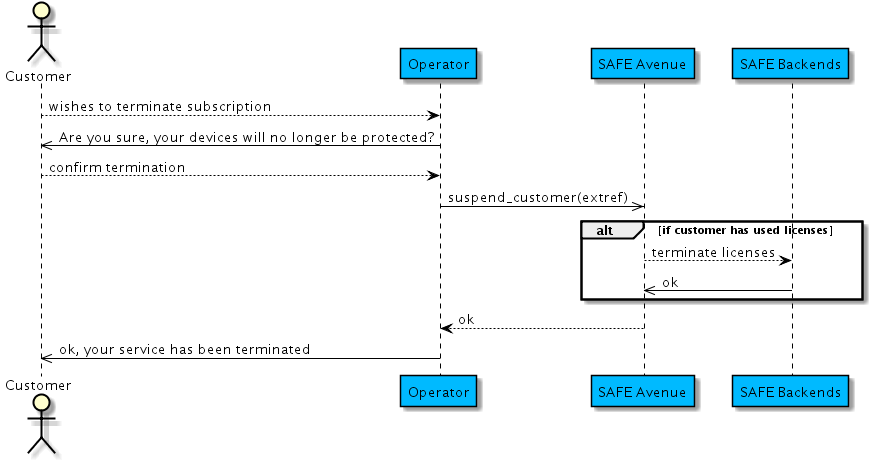
For this scenario, the mandatory parameter is:

* 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
* F-Secure Avenue API

**Integration methods:**

suspend\_customer()

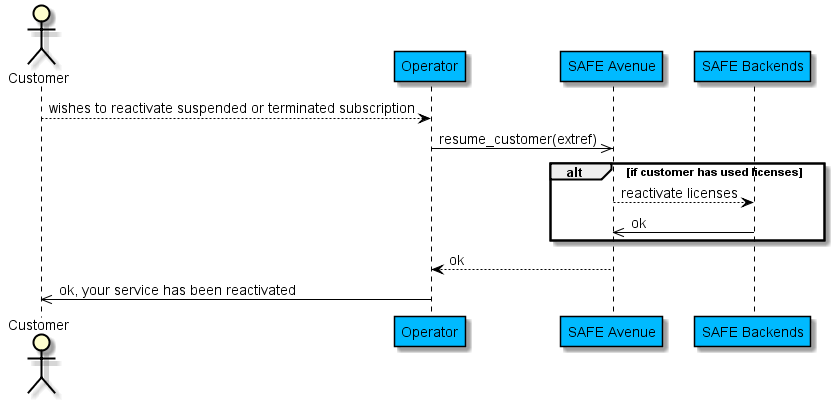
For this scenario, the mandatory parameter is:

* 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 emcustomer's SAFE subscription is terminated.

**Sequence diagram:**



**Involved components:**

* Operator CRM
* F-Secure Avenue API

**Integration methods:**

resume\_customer()

For this scenario, the mandatory parameter is:

* Extref: the external reference of the customer

## Client installation use cases

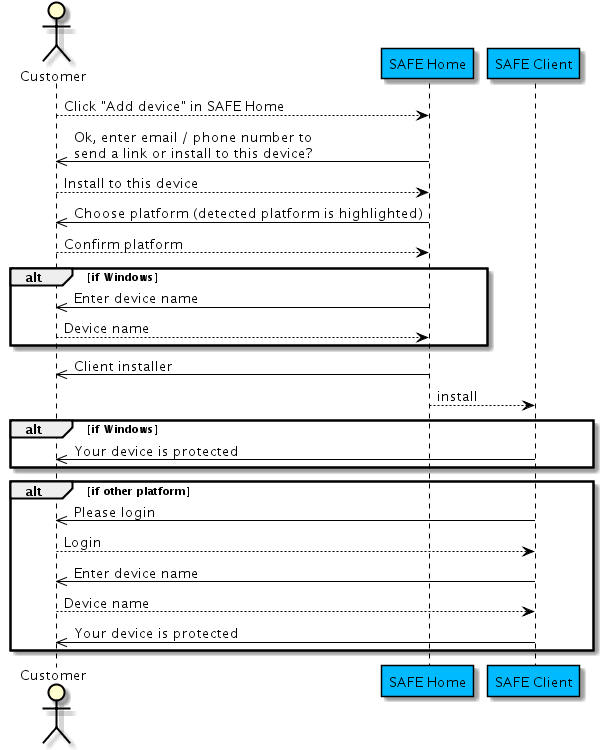
The Client installation use cases do not require any API integration from the operator, however it is important for an Operator to review them, as most of them are linked to customizable and cobrandable components, and understanding these use cases is imperative to understand the SAFE user journeys from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



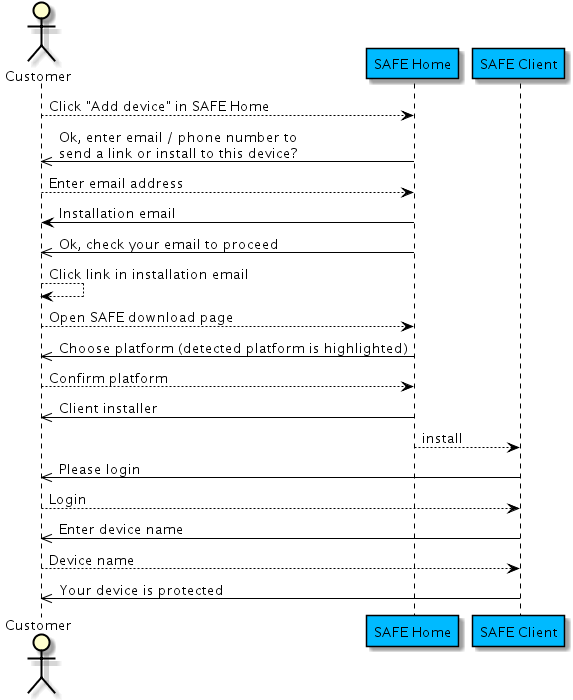
**Involved components:**

* SAFE Home
* SAFE Client
* Operator Identity Provider for Authentication
* AppStores (If downloading a mobile client, the download is served from the respective AppStore)

### 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:**



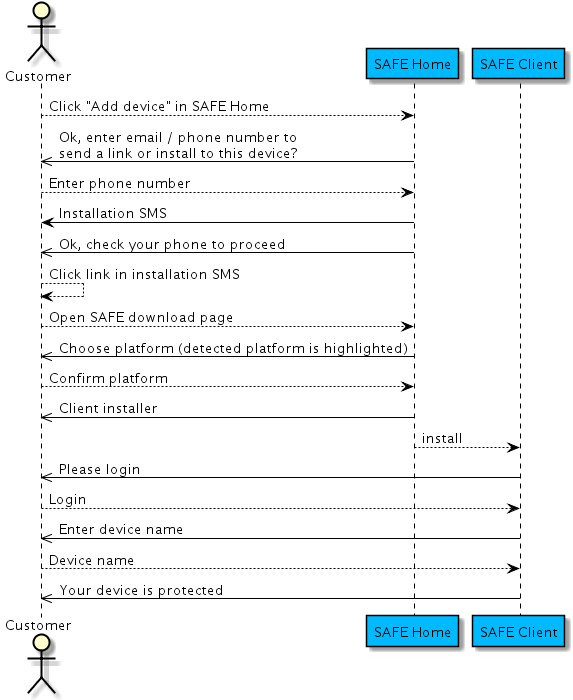
**Involved components:**

* SAFE Home
* SAFE Client
* Operator Identity Provider for Authentication
* 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:**



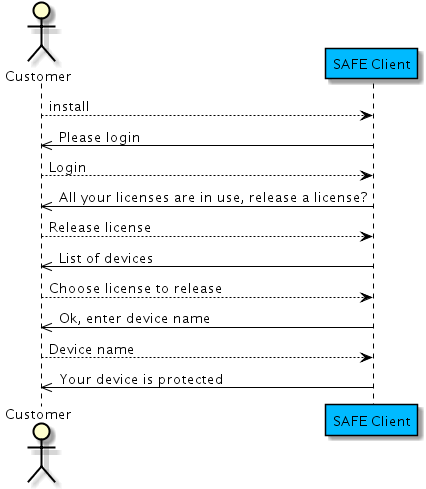
**Involved components:**

* SAFE Home
* SAFE Client
* Operator Identity Provider for Authentication
* 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:**

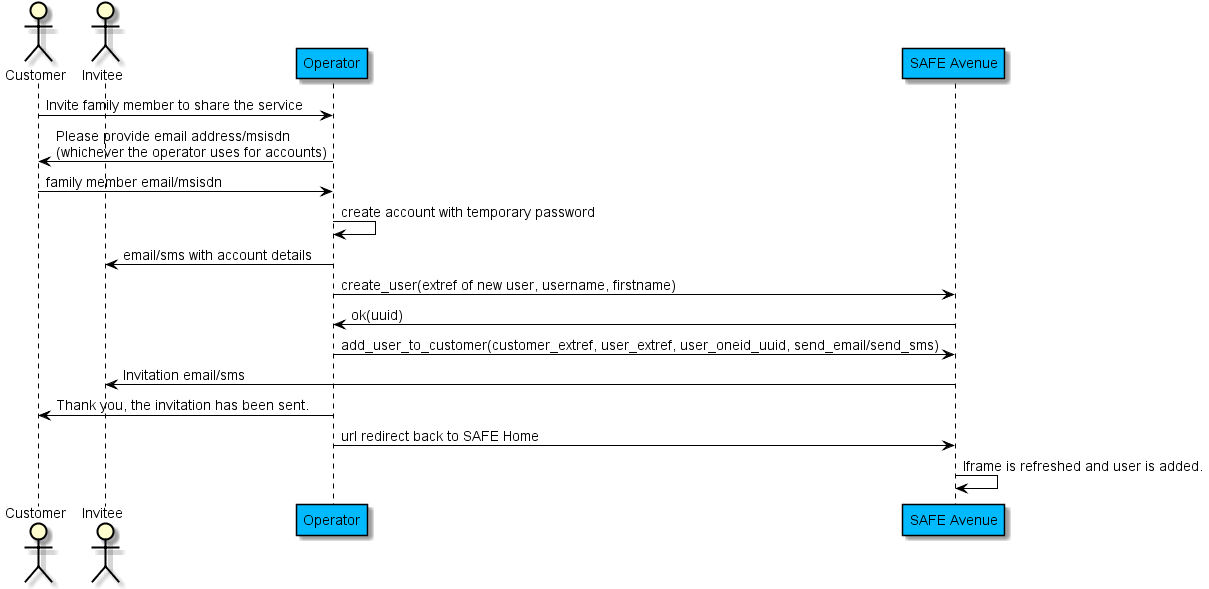
* SAFE Home
* SAFE Client

## Family use cases

### UCF-1: A Customer invites a family member to share the service

SYNOPSIS: A customer wants to a family member of friend to be able to protect his/her devices using SAFE, and invites them to the service. Add person in SAFE Home has been configured to point to operator form.   
EXPECTED END RESULT: The sub-user receives the invitation email and is able to start protecting his or her devices.

**Sequence diagram:**



 After this sequence, the sub-user is able to protect a device with the UCI-2 flow.

**Involved components:**

* Operator CRM
* Co-branding portal

# Deployment of the client applications

## **Mobile clients in AppStores**

It is imperative to deploy the end point applications to mobile platforms via the platforms respective application stores. Even though on some ecosystems it is technically possible to deploy applications as direct downloads of the installer binaries, doing so results in poor user experience and potentially reduces the security of the device as a result, a side effect that should not be associated with a security service.

Different application stores set different requirements to applications and the publishing process can be very different. F-Secure will help on submission related questions, but the actual submission is done by operator. More information can be obtained from [Application Submission guides](https://fscompass2.f-secure.com/pages/viewpage.action?pageId=4489528) in Compass.

# API integration

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). It is required that the Content-Type request header is set as application/x-www-form-urlencoded.

## Request Headers

Safe Avenue API uses standard HTTP requests headers for some procedures and custom HTTP request headers for special purposes.

### Content-Type

This is required to be set for POST procedures. Value is application/x-www-form-urlencoded.

### X-Transaction

This is for transaction identifier. By default, Safe Avenue generates its own transaction identifier. Operator may provide their own transaction identifier through this header. When provided, it will be appended to the transaction identifier generated by Safe Avenue. It shall be logged and can be used to debug events that has happened in Safe Avenue and related F-Secure systems.

* Usage is optional
* Consists of alphanumeric ASCII characters only
* Case-insensitive value
* Recommended length is 6 to 16 characters

## 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 successful 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"  
 },  
 "transaction\_id": "0000-w8nvqt3rhp7wgr20",  
 "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,  
 "transaction\_id": "0000-bgl4syshskw4dha7",  
 "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,  
 "transaction\_id": "0000-yst916htr3wh8w6q",  
 "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.

In the JSON content, it also has the transaction identifier, transaction\_id, that corresponds to the API request. In addition, the transaction identifier can also be obtained from the custom HTTP response header, X-Transaction.

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).

### Product Category type

Each Product falls under a ProductCategory.

ProductCategory = String

Below are the possible values for ProductCategory.

|  |  |
| --- | --- |
| ProductCategory | Description |
| "SECURITY" | Security product |
| "PRIVACY" | Privacy product |
| "PASSWORD" | Password product |

### Customer type

A Customer represents a group and group owner. A group consists of an owner and maximum eight subusers, in total maximum nine members. In Safe Avenue API, a group is referred to as a customer.

Each group member is a user, please refer to [user account](#user-account) for more info.

Customer = {  
 extref: String,  
 oneid\_uuid: String,  
 customer\_type: CustomerType,  
 license\_size: Size,  
 privacy\_license\_size: Size,  
 password\_license\_size: Size,  
 expiry\_date: DateTime,  
 customer\_identifier: String,  
 quota: Size,  
 total\_quota\_blocks: Size,  
 unused\_quota\_blocks: Size,  
 owner\_quota\_blocks: 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 user id for easy mapping between operator user 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}$. When using SSO, extref is also owner's user account external reference.

oneid\_uuid

A unique identifier that references the owner'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 security licenses that can be provisioned for the group members, maximum value allowed for license size is 256.

privacy\_license\_size

The number of privacy licenses that can be provisioned for the group members, maximum value allowed for license size is 256.

password\_license\_size

The number of password licenses that can be provisioned for the group members, 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 per block.

total\_quota\_blocks

Total quota blocks that customer (group) has, see [storage quota](#storage-quota).

owner\_quota\_blocks

Total quota blocks that is allocated to the customer (owner), see [storage quota](#storage-quota).

total\_unused\_blocks

Total unallocated quota blocks that customer (group) has, see [storage quota](#storage-quota).

An example of Customer object:

{  
 extref: 'ASDFZXCV',  
 license\_size: 7,  
 privacy\_license\_size: 5,  
 password\_license\_size: 5,  
 status: 'VALID',  
 customer\_identifier: 'ASDF-ZXCV-QWER-UIOP',  
 customer\_type: 'PREDEFINED',  
 expiry\_date: '2012-12-12T23:59:59Z',  
 quota: 10000,  
 total\_quota\_blocks: 10,  
 total\_unused\_blocks: 7,  
 owner\_quota\_blocks: 3,  
  
}

### Subuser type

A subuser is a member of a customer (group) who is not the group owner:

{  
 oneid\_uuid: String,  
 extref: String,  
 username: String,  
 customer\_identifier: String,  
 user\_quota\_blocks: Size  
}

oneid\_uuid

Unique identifier to subuser's [user account](#user-account).

extref

External reference to subuser's [user account](#user-account).

username

Username of the subuser's [user account](#user-account).

user\_quota\_blocks

Total quota blocks that is allocated to the subuser, see [storage quota](#storage-quota).

### Device type

Device platform:

DeviceType = String

Below are 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 member'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's owner:

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's owner?

### User info type

A Safe Avenue user information:

UserInfo = {  
 extref: String,  
 customer\_extref: String,  
 oneid\_uuid: String,  
 subuser: Boolean,  
 marketing\_permission: Boolean  
}

extref

User's extref. If using FSIDP, this value will be null.

customer\_extref

If user is a customer, customer\_extref will be the same as extref. If user is an active subuser, customer\_extref will be the group owner's extref. If user is an orphaned subuser (a subuser who does not belong to any group), customer extref will be null.

oneid\_uuid

A unique identifier that references the owner's [user account](#user-account).

subuser

Flag indicating if the user is a subuser. Subuser can be an active or orphaned subuser.

marketing\_permission

Flag indicating if user consented to marketing communications.

### 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 |
| "lt" | Lithuanian |
| "nl" | Dutch |
| "no" | Norwegian |
| "pl" | Polish |
| "pt" | Portuguese |
| "pt-br" | Brazilian Portuguese |
| "ru" | Russian |
| "sl" | Slovene |
| "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 security products that can be used to call [POST provision\_license()](#post-provision_license).

**No parameter**

**Return** list of products:

{  
 products : [Product]  
}

### POST send\_welcome\_email()

Send welcome email to customer.

This API can only be used if the customer is already named and created via API.

**Parameters**

extref: String

Extref of customer

**Returns** {}

**Errors**

does\_not\_exist

If customer does not exist.

customer\_not\_named

If customer is not named.

no\_user\_email\_addr\_for\_welcome\_email

The customer has no email address for sending welcome email.

### POST send\_welcome\_sms()

Send welcome SMS to customer.

This API can only be used if the customer is already named and created via API.

**Parameters**

extref: String

Extref of customer

**Returns** {}

**Errors**

does\_not\_exist

If customer does not exist.

customer\_not\_named

If customer is not named.

no\_user\_phone\_number\_for\_welcome\_sms

The customer has no phone number for sending welcome SMS.

### POST send\_download\_email()

Send an email to customer with instructions and link to product download page. One email can only have content on one ProductCategory.

The download email template can be customized if needed. Each ProductCategory, has its own email template.

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.

product\_category: ProductCategory

Optional, default "SECURITY". Product category for the email content.

**Returns** {}

**Errors**

invalid\_locale

If provided locale is invalid.

invalid\_product\_category

If provided product category 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. One SMS can only have content on one ProductCategory.

The download SMS template can be customized if needed. Each ProductCategory, has its own SMS template.

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. Format should be '+' followed by at least 8 digits.

locale: Locale

Locale of the SMS content.

product\_category: ProductCategory

Optional, default "SECURITY". Product category for the SMS content.

**Returns** {}

**Errors**

invalid\_locale

If provided locale is invalid.

invalid\_product\_category

If provided product category 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 assign a [user account](#user-account) to the customer as owner.

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(). See [user account](#user-account) for details.

**Parameters**

extref: String

Customer extref.

license\_size: Size

Optional, default 0. Security license size of the created customer.

privacy\_license\_size: Size

Optional, default 0. Privacy license size of the created customer.

password\_license\_size: Size

Optional, default 0. Password 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. Storage quota size per block.

total\_quota\_blocks: Size

Optional, default 1. Total quota blocks for customer (group), see [storage quota](#storage-quota).

owner\_quota\_blocks: Size

Optional, default 1. Number of quota blocks for customer (owner), see [storage quota](#storage-quota) size.

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

username: String

Optional. Required only if using SSO. Owner's username.

email\_addr: EmailAddress

Optional. Required only if using FSIDP and phone\_number is not specified. Owner's email address.

phone\_number: PhoneNumber

Optional. Required only if using FSIDP and email\_addr is not specified. Owner's mobile phone number.

first\_name: String

Owner's first name.

last\_name: String

Optional. Owner's last name.

user\_locale: Locale

Optional. Owner's locale.

send\_email: Flag

Optional, default 1. This flag is only applicable when email\_addr is specified. Send welcome email to owner after creating his user account?

send\_sms: Flag

Optional, default 1. This flag is only applicable when phone\_number is specified. Send welcome SMS to owner after creating his 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 or the user is already a member of another customer.

license\_limit\_reached

If the specified license\_size or privacy\_license\_size or password\_license\_size is greater than maximum license size. See [customer type](#customer-type).

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.

insufficient\_available\_quota\_blocks

If owner\_quota\_blocks is greater than total\_quota\_blocks, see [storage quota](#storage-quota).

username\_already\_used

See [user account](#user-account) about retrying create\_customer().

email\_address\_already\_used

See [user account](#user-account) about retrying create\_customer().

phone\_number\_already\_used

See [user account](#user-account) about retrying create\_customer().

error\_creating\_customer

See [user account](#user-account) about retrying create\_customer().

param\_mismatch

See [user account](#user-account) about retrying create\_customer().

extref\_mismatch

See [user account](#user-account) about retrying 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 and assign it as owner.

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

See [user account](#user-account) for more info.

**Parameters**

extref: String

Customer extref.

username: String

Optional if using FSIDP. Required if using SSO. Owner's username.

email\_addr: EmailAddress

Optional if using SSO. Required if using FSIDP and phone\_number is not specified. Owner's email address.

phone\_number: PhoneNumber

Optional if using SSO. Required if using FSIDP and email\_addr is not specified. Owner's mobile phone number.

first\_name: String

Owner's first name.

last\_name: String

Optional, owner's last name.

user\_locale: Locale

Optional, default to configured operator locale. Owner's locale.

send\_email: Flag

Optional, default 1. This flag is only applicable when email\_addr is specified. Send welcome email after creating owner's user account?

send\_sms: Flag

Optional, default 1. This flag is only applicable when phone\_number is specified. Send welcome SMS after creating owner's 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 (extref, email address, phone number, first name, last name) are invalid.

naming\_constraint\_error

If one of the following criterias is met:

* When send\_email = 1 but email\_addr is not specified
* When send\_sms = 1 but phone\_number is not specified
* When using FSIDP, email\_addr and phone\_number are specified but send\_email = 0 and send\_sms = 0.
* When using FSIDP, email\_addr is specified and phone\_number is not specified but send\_email = 0.
* When using FSIDP, phone\_number is specified and email\_addr is not specified but send\_sms = 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 or the user is already a member of another customer.

group\_user\_already\_exists

The user with same username already exists in operator group.

customer\_locked

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

### POST create\_user()

Create a user, see [user account](#user-account) for more info.

**Parameters**

extref: String

Mandatory if using SSO. Not required if using FSIDP. User extref.

username: String

Optional. Required if using SSO. User's username.

email\_addr: EmailAddress

Optional. Required only if using FSIDP and phone\_number is not specified. User's email address.

phone\_number: PhoneNumber

Optional. Required only if using FSIDP and email\_addr is not specified. User's mobile phone number.

first\_name: String

User's first name.

last\_name: String

Optional. User's last name.

locale: Locale

Optional, default to configured operator locale. User's locale.

**Returns** unique identifier of the created user:

{

oneid\_uuid: String

System Message: WARNING/2 (/opt/jenkins/jobs/handbook-deploy/workspace/doc/spec-v3.rst, line 1001)

Definition list ends without a blank line; unexpected unindent.

}

**Errors**

invalid\_locale

If user\_locale is invalid.

invalid\_user\_details

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

naming\_constraint\_error

If one of the following criterias is met:

* When using SSO but username is not provided.

user\_already\_exists

The user with same username already exists.

group\_user\_already\_exists

The user with same username already exists in operator group.

### POST delete\_user()

Delete a user. The user is identified by either extref or oneid\_uuid. This API call is permanent and there is no roll-back. See [user account](#user-account) for details.

**Parameters**

extref: String

Mandatory if using SSO or if oneid\_uuid is not provided. Not required if using FSIDP. User extref.

oneid\_uuid

Mandatory if extref is not provided. User unique identifier.

**Returns** {}

**Errors**

user\_does\_not\_exist

If user does not exists.

customer\_locked

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

incomplete\_update

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

incomplete\_group\_update

See [incomplete group update error](#incomplete-group-update-error).

### GET get\_user()

Get details of a user. The user is identified by oneid\_uuid.

**Parameters**

oneid\_uuid

Mandatory. User unique identifier.

**Returns** the queried user information:

{  
 user\_info: UserInfo  
}

**Errors**

does\_not\_exist

If the user does not exist.

### POST add\_user\_to\_customer()

Add the user into customer as subuser. The user is identified by either user\_extref or user\_oneid\_uuid but not both.

**Parameters**

user\_extref: String

User external reference.

user\_oneid\_uuid: String

User unique identifier.

customer\_extref: String

Customer external reference.

send\_email: Flag

Optional, default 1. This flag is only applicable if user has an email address. Whether to send invitation email to the user after the user is added to the customer. The user's locale is used to send the welcome email.

send\_sms: Flag

Optional, default 1. This flag is only applicable if user has a phone number. Whether to send invitation SMS to the user after the user is added to the customer. The user's locale is used to send the welcome SMS.

**Returns** {}

**Errors**

does\_not\_exist

The user or customer does not exist.

customer\_not\_named

The customer is not named.

already\_exists

The user is already added to this customer or another customer.

member\_count\_limit\_reached

Maximum number of members for this customer is reached.

no\_user\_email\_addr\_for\_welcome\_email

The user has no email address for sending welcome email if send\_email=1.

no\_user\_phone\_number\_for\_welcome\_sms

The user has no phone number for sending welcome SMS if send\_sms=1.

### POST remove\_user\_from\_customer()

Remove the user from customer. The user is identified by either user\_extref or user\_oneid\_uuid but not both.

**Parameters**

user\_extref: String

User external reference.

user\_oneid\_uuid: String

User unique identifier.

customer\_extref: String

Customer external reference.

**Returns** {}

**Errors**

does\_not\_exist

The user or customer does not exist or the user is not a member of this customer.

customer\_not\_named

The customer is not named.

### POST update\_user\_quota\_blocks()

Provision quota for a member of a customer. The user is identified by either extref or oneid\_uuid but not both.

**Parameters**

extref: String

User external reference.

oneid\_uuid: String

User unique identifier.

quota\_blocks: Size

Number of quota blocks to be allocated for the user, see [storage quota](#storage-quota) size.

**Returns** {}

**Errors**

customer\_not\_named

If the customer is not linked with a user account

does\_not\_exist

The user or customer does not exist or the user is not a member of this customer.

insufficient\_available\_quota\_blocks

If allocating quota\_blocks for the user will accumulate to customer's (group) used blocks to be greater than its total quota blocks, see [storage quota](#storage-quota).

customer\_locked

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

illegal\_state\_error

See [illegal state](#illegal-state).

### GET get\_customer()

Get details of a customer. The customer is identified by extref, oneid\_uuid or both. If both are specified, they need to be for the same customer.

**Parameters**

extref: String

Optional. Customer extref.

oneid\_uuid

Optional. Customer unique identifier.

email\_address

Optional. Customer email\_address.

phone\_number

Optional. Customer phone number.

include\_licenses: Boolean

Optional, default 0. Include licenses data in response?

include\_subusers: Boolean

Optional, default 0. Include subusers data in response?

**Returns** the queried customer, includes active security, privacy, and password licenses if include\_licenses = 1, includes subusers if include\_subusers = 1:

{  
 customer: Customer,  
 licenses: [License],  
 privacy\_licenses: [License],  
 password\_licenses: [License],  
 subusers: [Subuser]  
}

**Errors**

does\_not\_exist

If the customer does not exist.

### POST rename\_customer\_extref()

Rename a customer extref. When using SSO, this API will also update extref in [user account](#user-account).

**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](#illegal-state).

extref\_already\_used

When new extref is already used by an existing user account (only applies when using SSO).

### POST update\_user()

Update [user account](#user-account). information

**Parameters**

extref: String

Optional. Required only if oneid\_uuid is not specified User extref.

oneid\_uuid: String

Optional. Required only if extref is not specified User unique identifier.

email\_address: String

Optional. New email address.

phone\_number: String

Optional. New phone number.

marketing\_permission: Boolean

Optional. Flag if user consented to marketing communication?

**Returns** {}

**Errors**

user\_does\_not\_exist

If the user does not exist

username\_already\_used

If the username is already used by another user account.

email\_address\_already\_used

If the email address is already used by another user account.

phone\_number\_already\_used

If the phone number is already used by another user account.

invalid\_user\_details

If user details (email address, phone number) are invalid.

### 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 security license size.

privacy\_license\_size: Size

Optional. New privacy license size.

password\_license\_size: Size

Optional. New password license size.

customer\_type: CustomerType

Optional. New customer type.

expiry\_date: DateTime

Optional. New expiry date.

quota\_size: Size

Optional. New storage quota size per block.

total\_quota\_blocks: Size

Optional. New total quota blocks for customer (group), see [storage quota](#storage-quota).

force\_downgrade: Flag

Optional, default 0. Force downgrade means to terminate oldest licenses when a new license size set is less than the number of active licenses, to prevent license\_limit\_reached error from being raised. This parameter is applicable to security, privacy, and password license size. License size evaluation and termination are done for the same ProductCategory. This means only security licenses will be terminated when the new security license size exceeds the number of active security licenses, same goes for privacy and password licenses.

For example you have a customer with security license size 3 and provisioned 3 security 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 security license size to 2 will fail because there are 3 active security 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's security license size will be successfully set to 2 and the oldest security licenses (here only product1 license) are terminated to make sure number of active security licenses are not more than the total security 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 or privacy\_license\_size or password\_license\_size is less than existing license size and force\_downgrade = 0, or the specified license\_size or privacy\_license\_size or password\_license\_size is greater than maximum license size. See license\_size, privacy\_license\_size, and password\_license\_size in [customer type](#customer-type).

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.

invalid\_total\_quota\_blocks

If the new total quota blocks is less than the total quota blocks that are already allocated to members.

customer\_locked

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

incomplete\_update

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

incomplete\_group\_update

See [incomplete group update error](#incomplete-group-update-error).

illegal\_state\_error

See [illegal state](#illegal-state).

### 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).

incomplete\_group\_update

See [incomplete group update error](#incomplete-group-update-error).

illegal\_state\_error

See [illegal state](#illegal-state).

### 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).

incomplete\_group\_update

See [incomplete group update error](#incomplete-group-update-error).

illegal\_state\_error

See [illegal state](#illegal-state).

### 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()

**(DEPRECATED)**

Provision a security license for a customer.

If the customer type is predefined then the provisioned license has 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](#illegal-state).

### POST terminate\_license()

Terminate a customer's security 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).

**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.

invalid\_product\_category

If the license specified by license\_uuid does not have "SECURITY" as its product category.

customer\_locked

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

illegal\_state\_error

See [illegal state](#illegal-state).

### POST rename\_device()

Update device name of a security, privacy or password 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\_product\_category |
| 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 | phone\_number\_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 |
| 403 | member\_count\_limit\_reached |
| 403 | no\_user\_email\_addr\_for\_welcome\_email |
| 403 | no\_user\_phone\_number\_for\_welcome\_sms |
| 403 | insufficient\_available\_quota\_blocks |
| 403 | invalid\_total\_quota\_blocks |
| 404 | does\_not\_exist |
| 405 | method\_not\_allowed |
| 500 | incomplete\_update |
| 500 | incomplete\_group\_update |
| 500 | internal\_error |
| 503 | server\_busy |

## Common errors

Common errors that can be returned by all procedures:

### Internal error

internal\_error

Unexpected server error

### Permission denied error

permission\_denied

Current API user does not have permission to call the procedure

### Method not allowed error

method\_not\_allowed

Unrecognized or wrong HTTP method

### Parameters not match error

parameters\_not\_match

Invalid parameter or invalid combination of parameters

### Server busy error

server\_busy

The procedure cannot handle the request due to limited resources

### Customer locked error

customer\_locked

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

incomplete\_update

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.

### Incomplete group update error

incomplete\_group\_update

Due to some limitations of Safe Avenue, certain operations are not atomic. For example, update\_customer() might return incomplete\_group\_update error, leaving the customer and its members in inconsistent state. Client should repeat the procedure when encountering incomplete\_group\_update error until success, in this case call update\_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().

### Suspended customer

When a customer is suspended:

* The licenses are also suspended, a suspended license is like terminated license (see [POST terminate\_license()](#post-terminate_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

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.

When using FSIDP, it is required that either email address or phone number is specified. Username is optional and need not be the same as email address or phone number. Also, for first name and last name:

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

When using SSO, username is required and need not be the same as email address or phone number. First name and last name validation depends on operator's identity provider.

A user can be created with either:

* [POST create\_customer()](#post-create_customer) with user details or
* [POST name\_customer()](#post-name_customer) or
* [POST create\_user()](#post-create_user)

A customer (group) may or may not have an owner. For example:

create\_customer(extref='cust001')  
# Customer 'cust001' has no owner  
  
name\_customer(extref='cust001',  
 email\_addr='cust001@mailinator.com',  
 first\_name='Cust',  
 last\_name='001')  
# Now the customer has owner

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 assign owner in this way is not atomic, the user account might be created but assigning owner might fail. To prevent user\_already\_exists error on second call using same user account, POST create\_customer() accepts existing user account (username, email address and phone number). 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, email\_addr and phone\_number must be the same as the initial request  
create\_customer(extref='cust001',  
 email\_addr='cust001@mailinator.com',  
 first\_name='Customer',  
 last\_name='One')

Note that when retrying create\_customer() username, email address and phone number 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 is already linked with another customer, username\_already\_used (SSO and FSIDP), email\_address\_already\_used (FSIDP), or phone\_number\_already\_used (FSIDP) will be raised.

When a user is created from [POST create\_user()](#post-create_user), the user is not associated with any customer either as owner or subuser. The user can then become owner of an existing customer by calling [POST name\_customer()](#post-name_customer) or of a new customer by calling [POST create\_customer()](#post-create_customer).

An unassociated user can also become a subuser of a customer by calling [POST add\_user\_to\_customer()](#post-add_user_to_customer) but can be removed from the customer with [POST remove\_user\_from\_customer()](#post-remove_user_from_customer).

There are restrictions of how a user can be associated with customer:

* a user can be a member of only a customer
* owner cannot be removed from owned customer
* maximum number of members for a customer is nine

Note that when using [POST delete\_user()](#post-delete_user): - It is not possible to recover user after executing this API call. - In order to be able to recover, use [POST suspend\_customer()](#post-suspend_customer) instead. For example, when customer fails to pay subscription bills. - Deleting a user account belonging to a customer will terminate all licenses and deny access to all subusers in the group. - Deleting a user account belonging to a subuser will terminate all licenses provisioned by the subuser and deny access to the subuser in the group.

## Storage quota

Customer storage quota is the number of megabyte (not mebibyte) for a single block of quota. The customer (group) may have multiple blocks of quota where each block of quota can be allocated to any of its members. However, at least one block needs to be allocated to the customer (owner). The total number of blocks that the customer (group) has is defined by its total quota blocks. Only unused quota blocks can be allocated to members in the group.

Customer (owner) storage can be provisioned from [POST create\_customer()](#post-create_customer) and [POST update\_user\_quota\_blocks()](#post-update_user_quota_blocks). 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
* Total quota blocks is at least 1.
* Owner quota blocks is at least 1.
* Provisioning owner quota blocks will not accumulate the total used blocks to be greater than total quota blocks

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

Subuser storage can be provisioned from [POST update\_user\_quota\_blocks()](#post-update_user_quota_blocks). 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*
* Subuser is linked with a user account
* Total quota blocks is at least 1.
* Owner quota blocks is at least 1.
* Subuser quota blocks is at least 1.
* Provisioning subuser quota blocks will not accumulate the total used blocks to be greater than total quota blocks

Storage can only be provisioned for subusers if customer (owner) has storage provisioned. If not [POST update\_user\_quota\_blocks()](#post-update_user_quota_blocks) will encounter [illegal state](#illegal-state).

Subuser storage can be terminated from [POST update\_user\_quota\_blocks()](#post-update_user_quota_blocks). Subuser quota blocks can be set to 0 to achieve this.