

Contents

1	Authoring	3
2	Group feature	4
3	Backward compatibility	5
4	Introducing the Group concept	6
4.1	Booking system	6
4.2	Partitioning system	6
4.3	What's a Group?	6
4.4	Origin group	13
4.5	Transient group	14
5	Working with Groups	15
5.1	General settings	15
5.2	Making a partition of devices	16
5.3	Making a reservation of devices	18
5.4	Facing conflicts	20
5.5	Monitoring groups	21
5.6	Monitoring devices	23
5.7	Managing groups	25
5.8	Managing users	26
5.9	Managing devices	29
6	RESTful API (add-on).....	32
6.1	User management	32
6.1.1	Access tokens	32
6.1.2	Devices	33
6.2	Users management	33
6.2.1	Users.....	33
6.2.2	Groups quotas	35
6.2.3	Access tokens.....	35
6.2.4	Devices	36
6.3	Devices management	38
6.3.1	Devices	38
6.3.2	Groups	39
6.4	Groups management	41
6.4.1	Groups	41
6.4.2	Devices	43
6.4.3	Users.....	45
7	Links between device controlling's API and ADB remote connection.....	47
8	Using Swagger UI.....	48
9	System configuration	50
10	Database migration.....	51
11	Implementation details	52
11.1	Architecture	52
11.2	Utilities	53

11.3 Impacted files..... 53

1 Authoring

Company: Orange SA – <https://www.orange.com>

Author: Denis Barbaron – denis.barbaron@orange.com

Version: 1.0

Date: June 7, 2019

2 Group feature

The Group feature added into STF is made of:

- the **Group** concept on which rely a **booking system** and a **partitioning system**, involving new tabs in the GUI both for setting and monitoring actions on groups, devices and users
- an **administrator level** in addition of the native user one, with increased rights on users, devices and groups
- a **Contact Support** button allowing for instance to contact the administrator by email from the login page in order to get an account
 - Note: it requires that email client software has been configured on the local computer of the user!
- a **Logout** button allowing to close properly the web session with the STF server

Note the Group feature rely also on 50 new APIs described at section 6, knowing that some of them are not used at UI level as some “users” privileged APIs (i.e. administrator level) for devices and access tokens management because they would have required to refactor in depth the corresponding existing screens. However, these privileged APIs can be used by third party clients like an automated tester as Jenkins for instance.

3 Backward compatibility

The Group feature has been added into STF taking care to maintain the backward compatibility with third party client software using STF APIs.

Moreover, although several screens and tabs have been added to the native STF UI to serve the considered new feature, the backward compatibility is also maintained at UI level so that users can continue to use STF as before if they have no need to use the **booking & partitioning** systems.

Finally, although new built-in objects have been added into the rethinkdb database impacting existing tables, the backward compatibility is also maintained at this level.

4 Introducing the Group concept

The Group concept has been introduced into STF to enable two precious features named the **booking system** and the **partitioning system** described briefly hereafter.

4.1 Booking system

The role of the **booking system** is to allow the user to reserve a set of devices for a set of users during a determined time window (ex: from 3:00 am to 4:00 am during 5 days).

It means in particular that during the considered time window, only the considered users may view and then take control of the reserved devices.

In synthesis, this **booking system** is very useful to ensure the availability of a set of devices for a test campaign, or for automated tests triggered by a task scheduler like Jenkins, or simply for a demo!

4.2 Partitioning system

For its part, the role of the **partitioning system** is to allow the administrator to create distinct sets of devices, each of them being viewable by a set of users without time limit.

The corollary is that each user may view zero, one or more of these sets which represent his own universe sharable or not or in part with the other users.

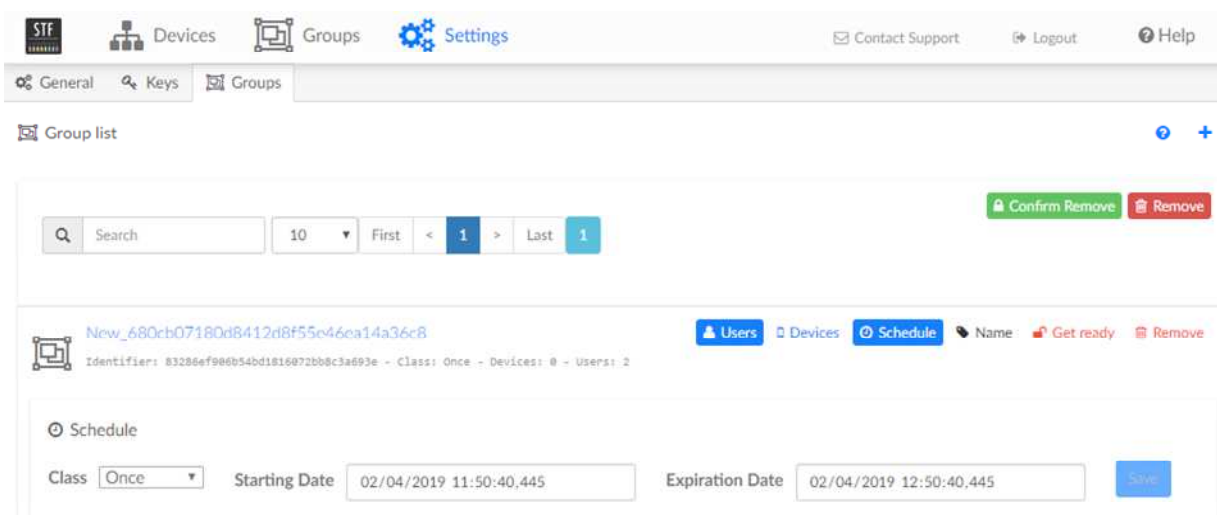
Note that without such a feature, all users share the same and unique universe of devices, which are all devices (i.e. legacy behaviour of STF).

Inside his universe, each user can then use the **booking system** to reserve one or more devices inside one or more reservations!

In synthesis, this **partitioning system** is very useful to manage inside a same physical STF platform a global set of devices allocated to different projects or organizations. In short, it is like to have a plurality of virtual and isolated STF platforms managed by a unique administrative entity.

4.3 What's a Group?

As suggested in the previous sections, which is common to the **booking & partitioning systems** are the association of users, devices and a specification of time, and this is precisely the general definition of a **Group**!



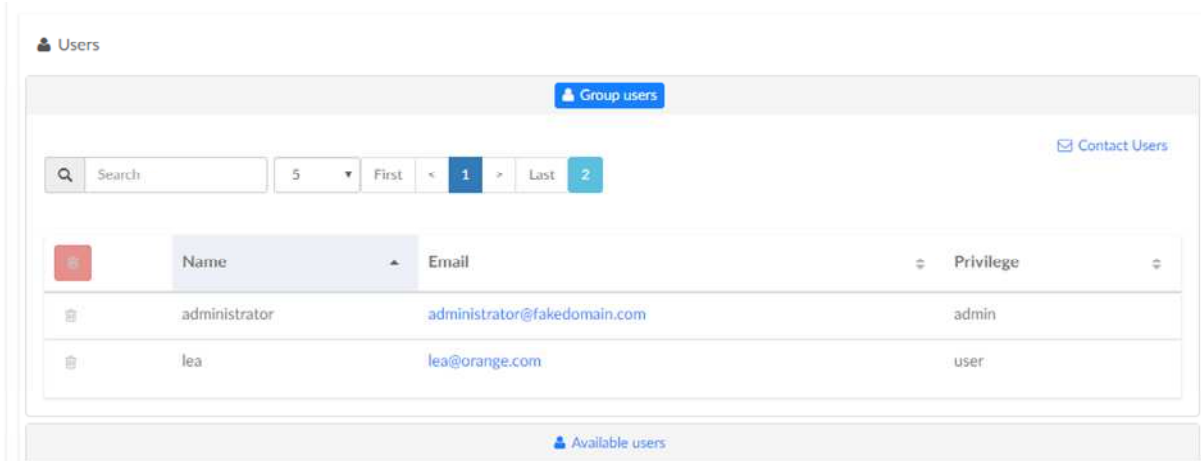


Figure 1: example of group creation

The above Figure 1 illustrates a typical creation of a group by a simple user (i.e. **lea** user).

The creation of a group is made by clicking on the **+** button!

Note to remove one particular group you have to click on the associated **Remove** button, and to remove a selection of groups (i.e. using the search field) you have to click on the global **Remove** button. In either cases you can activate or not a confirmation window before removing by clicking respectively on the **Confirm Remove** button or on the **Confirm Remove** button.

Note also **lea** has a quota for group creation and device booking which is managed by the administrator user, meaning in case her quota is reached an error message is returned on creation action. You may report to section 5.8 for details about quotas managing.

At creation time, the group has some default properties:

- it has a name which is randomly generated (i.e. New_...),
- it has an unique Identifier (e.g. for API use),
- it has currently no devices and two built-in users: the owner of the group (i.e. **lea**) and the administrator user who has the same rights as **lea** on the group (i.e. update, deletion),
- the schedule is the specification of time for which the considered users may view and take control of the considered devices: one time from creation time for one hour.

For now, let's change the name of the group and its schedule by clicking respectively on the **Name** and **Schedule** buttons!

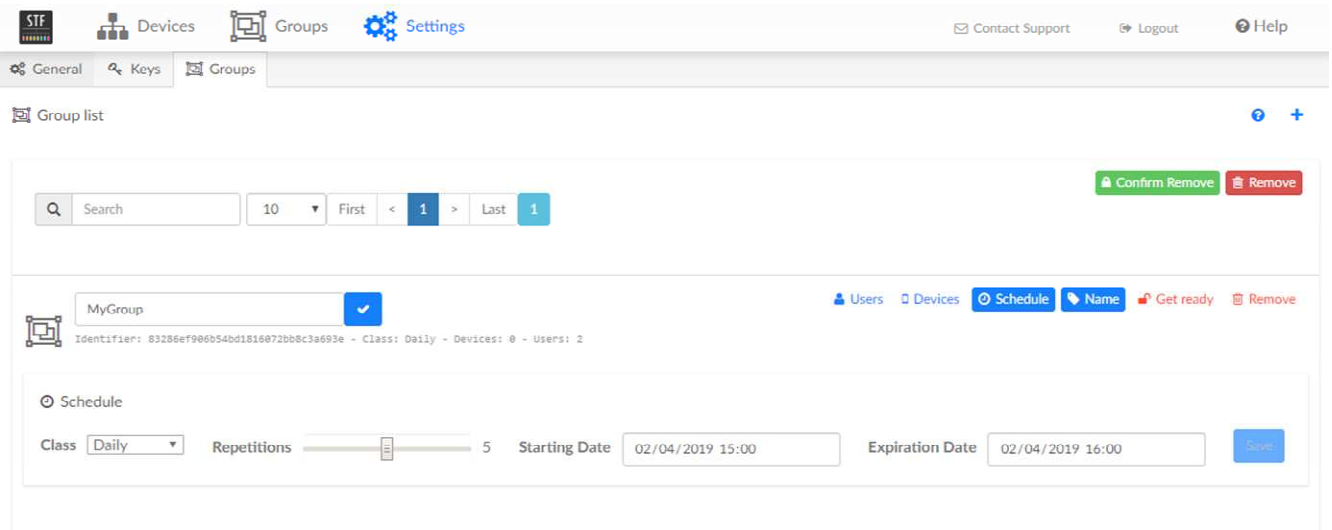


Figure 2: group name & schedule setting

As shown in Figure 2, some group properties have been changed:

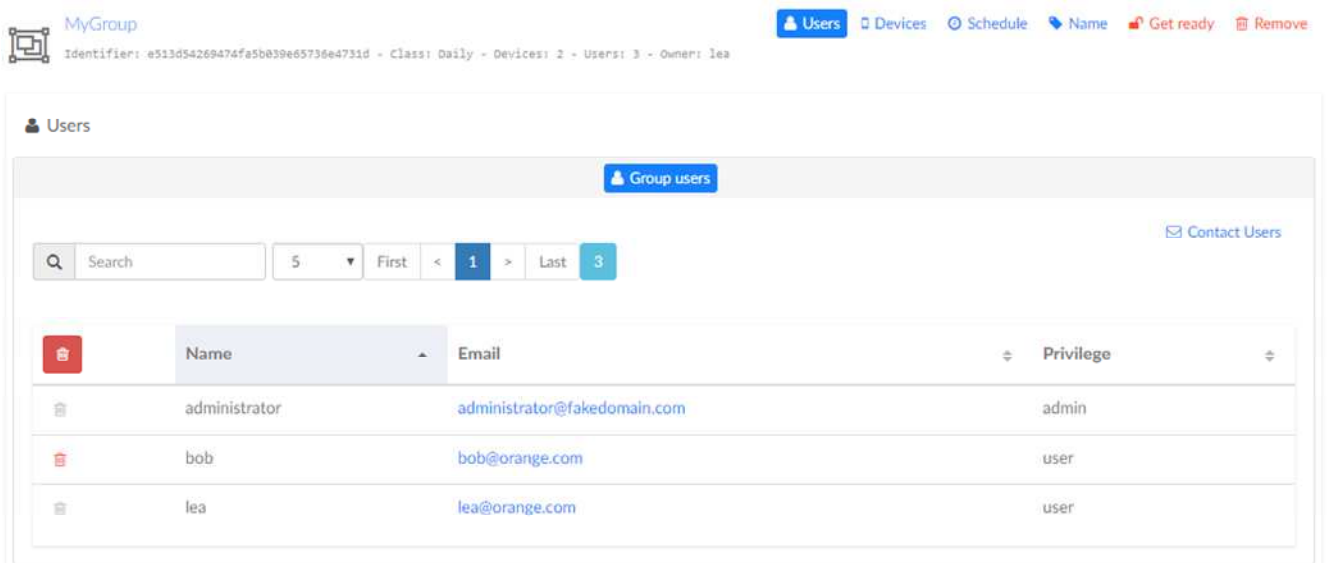
- the group name: **MyGroup**,
- the group schedule: from 3:00 pm to 4:00 pm each day during 6 days (i.e. one time + 5 repetitions).

Note the group schedule is characterized by a class that determines both the type of the group and the properties of the associated time window: periodicity, repetitions and duration.

There are two types of groups: **origin** and **transient**, these two concepts are described more in detail in sections 4.4 and 4.5.

MyGroup is a **transient** group which represents typically a reservation of devices for a determined time, managed by the **booking system**, while an **origin** group is part of the **partitioning system** designed to build a stable device universe for each user (i.e. administrator level).

Now let's add **bob** user into **MyGroup** by clicking on the **Users** button!



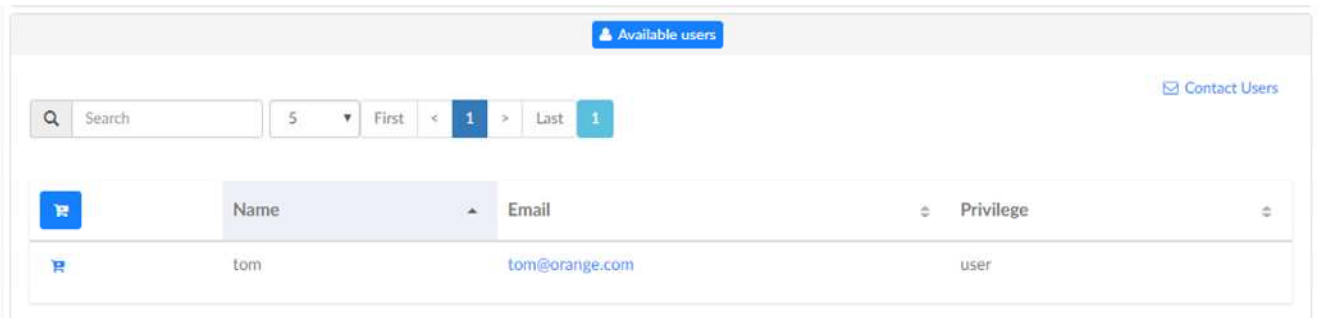






Figure 3: add users to a group

As shown in the Figure 3, the **bob** user has been added into **MyGroup** as a guest by clicking on the associated cart button , meaning he will be allowed to access the group devices during the active time of **MyGroup**, but without edition rights on **MyGroup**.

Of course, you may also add a selection of users (i.e. using the search field) into **MyGroup** by clicking on the selection cart button .

Conversely you may also remove user(s) from **MyGroup** using the corresponding trash buttons  and , except the owner of the group as well as the administrator user which are built-in users for **MyGroup** meaning they can't be removed!

Note in **Available users** tab appear all available users registered into the STF platform, which allows **lea** to invite not only users which have the same universe of devices as her, but also all other users like **bob**, which is a very useful way to lend some devices to users for a determined time.

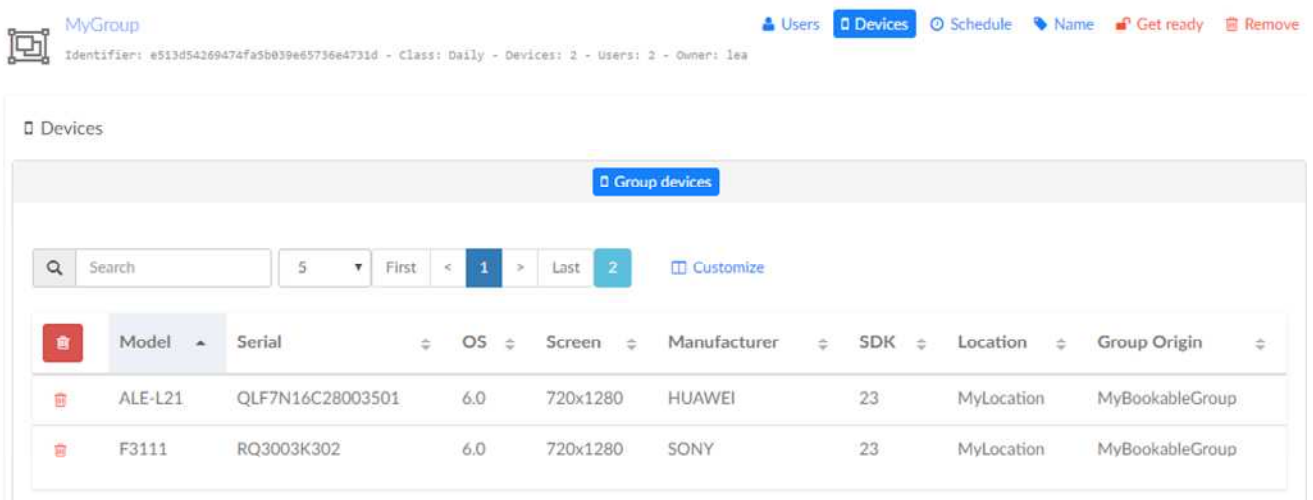
In our example, we suppose that **bob** and **lea** have distinct device universes, while **tom** and **lea** share the same one.

For his part, the administrator user enjoys all the rights on all groups, which means obviously his device universe matches all devices registered into the STF platform.

We shall see later what are the consequences of this for these users on their devices view during the lifecycle of **MyGroup**.

Also, let's remember that, as introduced in section 4.2, the device universe of each user may be made of zero, one or more **origin** groups, along each user may belong to zero, one or more **transient** groups.

But for now let's add some devices into **MyGroup**!



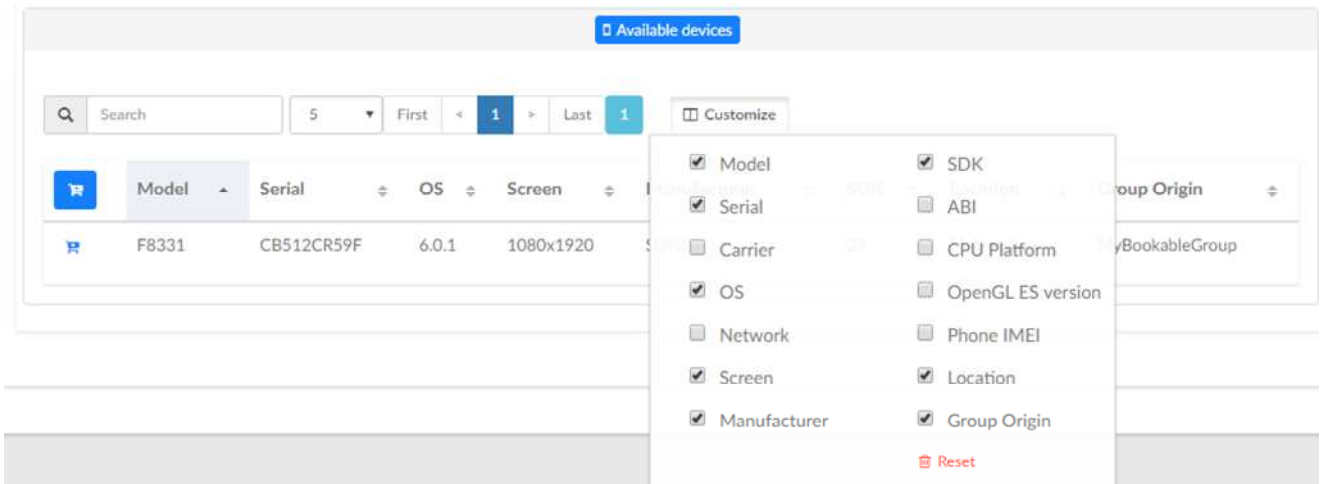




Figure 4: add devices to a group

As shown in the above Figure 4, two devices have been added into **MyGroup** by clicking on the associated cart button  : **ALE-21** and **F311** models.

Of course, you may also add a selection of devices (i.e. using the search field) into **MyGroup** by clicking on the selection cart button .

Conversely you may also remove device(s) from **MyGroup** using the corresponding trash buttons  and .

Note in **Available devices** tab appear only the bookable device universe of **lea** (i.e. **MyBookableGroup**), restricted to the devices which are not yet booked elsewhere in a time intersecting with the schedule of **MyGroup** in order to avoid any conflict!

In other words, it means a device may be booked by more than one transient group provided that there is no overlapping of the corresponding group time windows.


So, now one wonders to which group belong the two added devices: the origin group named **MyBookableGroup** owned by the administrator user or the transient group named **MyGroup** owned by **lea**, or both perhaps?

And therefore, at what time which users may access to these two devices?

In fact, the rule is that at a given time a device has always one origin group and one current group associated, and only users belonging to the current group of a device may access to it!

As explained in section 4.4, a device is assigned to an origin group either automatically at discover time by the system or manually by the administrator user, knowing that an origin group is always active and a device may belong to one origin group only.

On the other hand, the current group of a device is a more dynamic setting; it means its value changes depending of the status value of the groups to which the device belongs over time.

For example, during the edition step of **MyGroup** schedule shown at Figure 2, meant by the  **Get ready** button exposed, **MyGroup** has the **pending** state (i.e. red color convention) meaning it is not yet scheduled by the system.

During this **pending** state, the current group of each of our three devices is either equal to their origin group, that is **MyBookableGroup**, or equal respectively to a third party transient group necessarily in **active** state (i.e. green color convention).

In both cases, according to the previous rule it means during this state the group devices are not accessible by the group users through **MyGroup** identity.

Rather, the considered devices are accessible for each of them by the users of the **active** group which they belong at a given time through the corresponding group identity.

Once the edition is fixed, the user may click on the **Get ready** button, afterwards the group state changes to the **ready** value (i.e. orange color convention), and it becomes scheduled by the system.

Note once **MyGroup** leaves the **pending** state, no further backward is possible, involving the group schedule and the group name edition is no more possible, while nevertheless it is always possible to add or remove users and devices into/from **MyGroup**.

The below Figure 5 shows **MyGroup** in **ready** state!

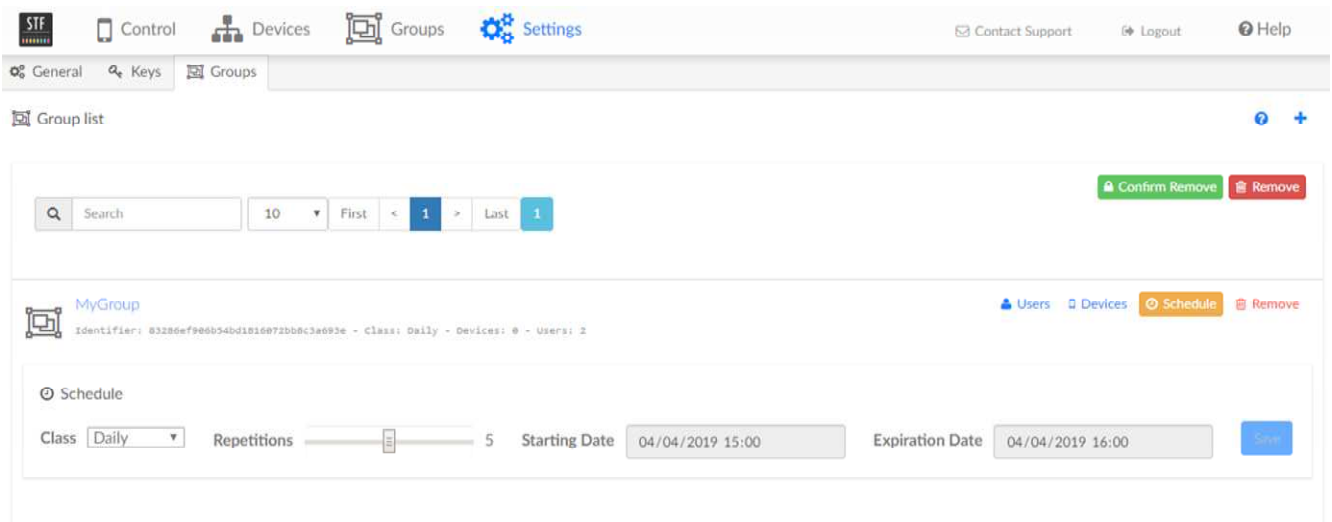


Figure 5: group in ready state

During **ready** state, meaning the current time is outside the group time window, the current group of the group devices is not equal to **MyGroup** and therefore the considered devices are not accessible by the group users through **MyGroup** identity.

Rather, the considered devices are accessible for each of them by the users of the **active** group which they belong at the given time through the corresponding group identity.

But when the current time becomes inside the group time window, the group state changes to the **active** value (i.e. green color convention), meaning the current group of the group devices is equal to **MyGroup** and therefore the considered devices become accessible by the group users through **MyGroup** identity.

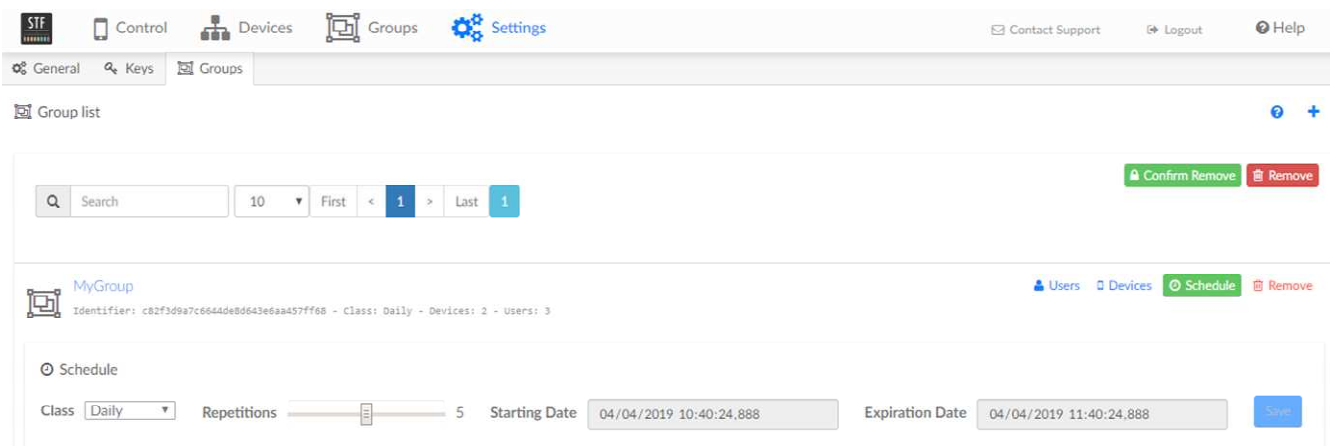


Figure 6: group in active state

The above Figure 6 shows **MyGroup** in **active** state!

So, now let's try to examine in practice what happens during the state transitions of **MyGroup**, in particular concerning device accessibility and device controlling from users?

During **ready** to **active** state transition of **MyGroup**, the latter becomes the current group of its two devices (i.e. ALE-21 and F311 models) meaning only **lea**, **bob** and the administrator user may access to them, excluding as a result **tom**!

Note that during the **active** state of **MyGroup**, the device universe of **bob** contains **MyGroup** devices in addition of **Common** devices!

So, you should know that the device universe of a user may fluctuate over time because it is made of the union of all devices that belong to the **active** groups which the user belongs (i.e. not only the origin groups).

But, what happens if at transition time **tom** was controlling the ALE-21 device? Well, in such case, **tom** would be simply got kicked out of the device in order to make it available for the users of **MyGroup**, it's the principle of a reservation!

This may sound brutal, but the truth is **tom** knew he used a **bookable** group device with the risk to be pre-empted by a **transient** group.

In order to avoid such situation, **tom** should have to make his own device reservation, or should have to use instead a **standard** group device ensuring to not be pre-empted by a third party group. You may report to section 4.4 for detailed explanations about bookable/standard group concepts.

On the other hand, if at this time **lea** was controlling the ALE-21 device, no kicking would have been performed simply because **lea** is also part of **MyGroup**!

Now, during **active** to **ready** state transition of **MyGroup**, that is when the current time becomes outside the group time window (i.e. repetition done), the origin group (i.e. **MyBookableGroup**) of our two considered devices (i.e. ALE-21 and F311 models) becomes their current group meaning **lea**, **tom** and the administrator user may access to them, excluding as a result **bob**!

But, what happens if at transition time **bob** was controlling the ALE-21 device? Well, in such case, **bob** would be simply got kicked out of the device in order to make it available for the users of **MyBookableGroup** of which he is not a member.

On the other hand, if at this time **lea** was controlling the ALE-21 device, no kicking would have been performed simply because **lea** is also part of **MyBookableGroup**!

In synthesis concerning the controlling/kicking of a device, the rule is that if the current group of a device is changing while the device is controlled by a user, the latter is then kicked out of the device if and only if the user is not a member of the new current group of the device.

Note the previous rule applies regardless of the type of the current group (i.e. transient or origin).

Now, one wonders if it is possible the origin group of a device may change while it is booked somewhere?

Yes, provided that the targeted origin group is also a bookable group and provided that the group owners of transient groups where the device is booked are also the members of the targeted origin group.

Otherwise, there would be a consistency issue because we will face to a situation in which a user could book a device which is not part of his device universe!

After this explanation about the Group concept, you may report to the section 5 titled "Working with Groups" to learn concretely to use groups to fill your requirements either as a simple user or an administrator user.

4.4 Origin group

An origin group has either a class attribute set to bookable or standard, and only the administrator is able to create such a group, because this is part of the **partitioning system** and by definition only the administrator is allowed to create universes of devices for the terminal users.

The screenshot displays two origin group configuration panels. The top panel is for the 'B_Common' group, which is of class 'Bookable'. It shows a starting date of 27/09/2018 11:11:34,644 and an expiration date of 27/09/2019 11:13:34,644. The bottom panel is for the 'Common' group, which is of class 'Standard'. It shows a starting date of 26/09/2018 18:40:34,256 and an expiration date of 26/09/2019 18:40:34,256. Both panels include a 'Schedule' section with a 'Class' dropdown, 'Starting Date' and 'Expiration Date' input fields, and a 'Save' button. The top panel also has 'Users' and 'Devices' icons and a 'Remove' button.

Figure 7: origin group setting

The purpose of the **bookable** class is to enable users to make reservations of devices, while the **standard** class is the opposite in order to keep the legacy behaviour of STF, meaning that if a user has taken control of a device belonging to a **standard** group, he is ensured that no reservation will pre-empt his device controlling.

Note at initialization time of STF database (cf. section 10), the non-removable administrator user is created as well as only one non-removable **standard** group (i.e. said as root group) owned by the administrator (e.g. **Common** group in Figure 7).

Each time a new device is detected by STF, it is by default automatically assigned to this group in order to preserve the legacy device controlling behaviour of STF.

So, it means that prior to enable users to make reservations, the administrator has to create at least one bookable group (e.g. **B_Common** group in Figure 7) and re-assign manually some devices to this group!

Note once a **bookable** or **standard** class is saved for a group, this last becomes immediately active, meaning the group name & schedule are no more editable!

Origin groups have also an infinite lifetime, except if they are explicitly deleted by the administrator.

It means that whatever the duration specified during their setting step, this duration is renewed as soon as the current end of life time is reached, for instance: $[t_0-t_{10}]$ then $[t_{10}-t_{20}]$ then $[t_{20}-t_{30}]$ etc, so that origin groups are always active!

In other words, an origin group is repeated infinitely with the same duration until its deletion.

How to add or remove one or more devices into/from an origin group is the same as with a transient one, except **Available devices** tab shows all devices of the STF platform for a bookable group, and it shows all not yet booked devices of the STF platform for a standard group.

4.5 Transient group

A transient group is part of the **booking system**, so any user may create such a group in order to make a reservation of devices, providing that some devices are available for booking of course!

The class attribute distinguishes two types of transient groups: periodic groups and non-periodic groups.

The below Figure 8 illustrates these two types of groups.

The screenshot displays two panels for configuring transient groups. The top panel is for 'My Daily group' (Identifier: daeb6cb291b84983b96868f951c35e80). It shows a 'Schedule' section with a 'Class' dropdown set to 'Daily', a 'Repetitions' slider set to 7, a 'Starting Date' of 19/03/2019 11:08:06.393, and an 'Expiration Date' of 19/03/2019 12:08:06.393. The bottom panel is for 'My Once group' (Identifier: 8671bd6bf28b44639c97f38caa81f3b8). It shows a 'Schedule' section with a 'Class' dropdown set to 'Once', a 'Starting Date' of 20/03/2019 11:08:37.653, and an 'Expiration Date' of 23/03/2019 12:08:37.653. Both panels include navigation buttons for 'Users', 'Devices', 'Schedule', 'Name', 'Get ready', and 'Remove'.

Figure 8: transient group setting

A non-periodic group as **My Once group** has the **once** class value, meaning it may be active during the specified duration only, so without any repetition.

A periodic group as **My Daily group** has a class value in the following set: **Hourly, Daily, Weekly, Monthly, Quaterly, Halfyearly, Yearly** and **Debug**, meaning it may be active during the specified duration using the specified periodic class value and the specified repetition number.

Note **Debug** class is reserved for administrator level only; it corresponds to a period of 5 minutes and should be used for test purpose only.

Unlike an origin group, a transient group is not eternal, it means at the end of its last repetition it is automatically cleaned by the system, so without requiring any user action.

5 Working with Groups

5.1 General settings

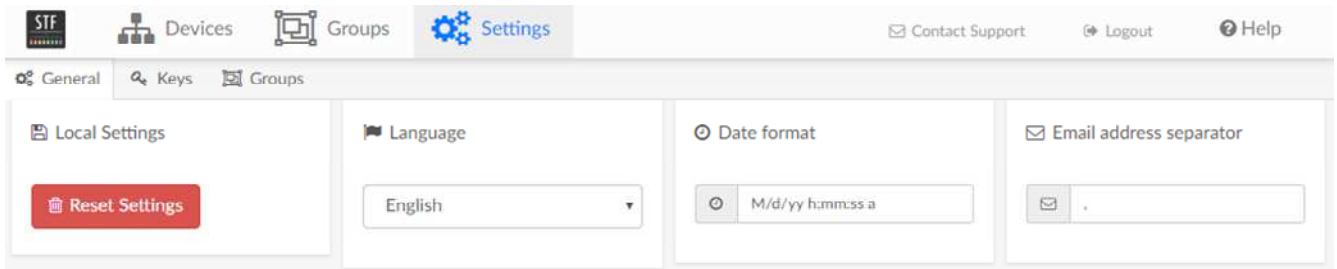


Figure 9: general settings

As shown in above Figure 9, you have to choose your own **Date format** to display the date values in global Devices & Groups views, following the below rules (i.e. extract from AngularJS API reference):

Formats date to a string based on the requested format.

format string can be composed of the following elements:

- **'yyyy'**: 4 digit representation of year (e.g. AD 1 => 0001, AD 2010 => 2010)
- **'yy'**: 2 digit representation of year, padded (00-99). (e.g. AD 2001 => 01, AD 2010 => 10)
- **'y'**: 1 digit representation of year, e.g. (AD 1 => 1, AD 199 => 199)
- **'MMMM'**: Month in year (January-December)
- **'MMM'**: Month in year (Jan-Dec)
- **'MM'**: Month in year, padded (01-12)
- **'M'**: Month in year (1-12)
- **'LLLL'**: Stand-alone month in year (January-December)
- **'dd'**: Day in month, padded (01-31)
- **'d'**: Day in month (1-31)
- **'EEEE'**: Day in Week,(Sunday-Saturday)
- **'EEE'**: Day in Week, (Sun-Sat)
- **'HH'**: Hour in day, padded (00-23)
- **'H'**: Hour in day (0-23)
- **'hh'**: Hour in AM/PM, padded (01-12)
- **'h'**: Hour in AM/PM, (1-12)
- **'mm'**: Minute in hour, padded (00-59)
- **'m'**: Minute in hour (0-59)
- **'ss'**: Second in minute, padded (00-59)
- **'s'**: Second in minute (0-59)
- **'sss'**: Millisecond in second, padded (000-999)
- **'a'**: AM/PM marker
- **'Z'**: 4 digit (+sign) representation of the timezone offset (-1200-+1200)
- **'ww'**: Week of year, padded (00-53). Week 01 is the week with the first Thursday of the year
- **'w'**: Week of year (0-53). Week 1 is the week with the first Thursday of the year
- **'G'**, **'GG'**, **'GGG'**: The abbreviated form of the era string (e.g. 'AD')
- **'GGGG'**: The long form of the era string (e.g. 'Anno Domini')

format string can also be one of the following predefined [localizable formats](#):

- **'medium'**: equivalent to **'MMM d, y h:mm:ss a'** for en_US locale (e.g. Sep 3, 2010 12:05:08 PM)
- **'short'**: equivalent to **'M/d/yy h:mm a'** for en_US locale (e.g. 9/3/10 12:05 PM)

- `'fullDate'`: equivalent to `'EEEE, MMMM d, y'` for en_US locale (e.g. Friday, September 3, 2010)
- `'longDate'`: equivalent to `'MMMM d, y'` for en_US locale (e.g. September 3, 2010)
- `'mediumDate'`: equivalent to `'MMM d, y'` for en_US locale (e.g. Sep 3, 2010)
- `'shortDate'`: equivalent to `'M/d/yy'` for en_US locale (e.g. 9/3/10)
- `'mediumTime'`: equivalent to `'h:mm:ss a'` for en_US locale (e.g. 12:05:08 PM)
- `'shortTime'`: equivalent to `'h:mm a'` for en_US locale (e.g. 12:05 PM)

`format` string can contain literal values. These need to be escaped by surrounding with single quotes (e.g. `"h 'in the morning'"`). In order to output a single quote, escape it - i.e., two single quotes in a sequence (e.g. `"h 'o' 'clock'"`).

Any other characters in the `format` string will be output as-is.

A default setting is proposed in the placeholder of the field (i.e. `M/d/yy h:mm:ss a`), which you should customize to fit your preferences!

You have also to choose the **Email address separator** required by your favorite email client launched by some introduced features.

Most of email clients require the comma character (i.e. default setting), but some other not as **Outlook** which requires the semi-colon character.

Note it is also possible to configure **Outlook** options to accept in addition the comma character as separator.

5.2 Making a partition of devices

Privileged feature: you have to login with administrator credentials to perform such operation.




The main reasons for device partitioning are either to dedicate a set of devices to a project or an organization, or to provide a set of devices for booking, or both.

In our use case, we would like to do both!

In fact, by default on a STF platform, all users and devices belong to the **Common** standard group meaning no device reservation is possible.

So, if we want to give the availability for **lea** and **tom** users to make some device reservations, we need to create at least a device partition dedicated to that.


Selecting the **Settings->Groups** tab, we have first to create a **bookable** group named **MyBookableGroup**:

-  : create a new group,
-  **Name** : modify its name to **MyBookableGroup** and save it,
-  **Schedule** : modify its class schedule to **bookable** and save it (i.e. this will also get ready the group!)

Now that **MyBookableGroup** is created and is active, it is possible to add some devices into it:

-  **Devices** : add **ALE-21** and **F3111** models

Finally, we have to add some users into it:

-  **Users** : add **lea** and **tom**

The Figure 10 below shows the result of the previous actions.

The screenshot displays the 'MyBookableGroup' interface. At the top, there are navigation buttons for 'Users', 'Devices', 'Schedule', and 'Remove'. Below this, a 'Schedule' section shows a class set to 'Bookable', a starting date of '2019-03-21T11:29:13.743', and an expiration date of '2020-03-21T12:29:13.743'. The main content is divided into three sections: 'Devices', 'Users', and 'Available devices'. The 'Devices' section shows a table with columns for Model, Serial, OS, Screen, Manufacturer, SDK, Location, and Group Origin. It lists two devices: ALE-L21 (Huawei) and F3111 (Sony). The 'Available devices' section shows a table with the same columns, listing one device: F8331 (Sony). The 'Users' section is divided into 'Group users' and 'Available users'. 'Group users' lists three users: administrator (admin), lea (user), and tom (user). 'Available users' lists one user: bob (user).

Figure 10: creation of a bookable group

So, the device universe of **lea** and **tom** is always the same except now they are able to use the booking system to reserve the **ALE-21** and **F3111** models.

On the other hand, the device universe of **bob** has been restricted to the **F8331** model without always having the ability to make any reservation.

Nevertheless, as illustrated in section 5.3, **bob** could be invited by the owner of a transient group (e.g. **lea**) to enjoy a device reservation.

Now, let's take a look on below Figure 11 showing the **Common** standard group state after this partitioning operation!

Common Users Devices Schedule Remove

Identifier: ee87ba65ce1443aa8e2cd7d77599fa99 - Class: Standard - Devices: 1 - Users: 4 - Owner: administrator

Schedule

Class: Standard Starting Date: 20/03/2019 11:18:35,954 Expiration Date: 19/03/2020 11:18:35,954 Save

Devices

Group devices

Search [] 5 First < 1 > Last 1 Customize

Model	Serial	OS	Screen	Manufacturer	SDK	Location	Group Origin
F8331	CB512CR59F	6.0.1	1080x1920	SONY	23	MyLocation	Common

Available devices

Search [] 5 First < 1 > Last 2 Customize

Model	Serial	OS	Screen	Manufacturer	SDK	Location	Group Origin
ALE-L21	QLF7N16C28003501	6.0	720x1280	HUAWEI	23	MyLocation	MyBookableGroup
F3111	RQ3003K302	6.0	720x1280	SONY	23	MyLocation	MyBookableGroup

Users

Group users Contact Users

Search [] 5 First < 1 > Last 4

Name	Email	Privilege
administrator	administrator@fakedomain.com	admin
bob	bob@orange.com	user
lea	lea@orange.com	user
tom	tom@orange.com	user

Available users



 No available users

Figure 11: the Common standard group

Note the **Group devices** tab contains the **F8311** model only, which explained why the device universe of **bob** has reduced, whereas the one of **lea** and **tom** has not changed, simply because they belong to both **Common** and **MyBookableGroup** groups.

Now that we have a ready to use bookable environment, let's go to the next section to learn about device reservation!

5.3 Making a reservation of devices






In our use case, **lea** is a developer on android mobile and she would like to invest five days to the development of her android application named **My Orange**.


Group feature for STF by Orange SA – Version 1.0

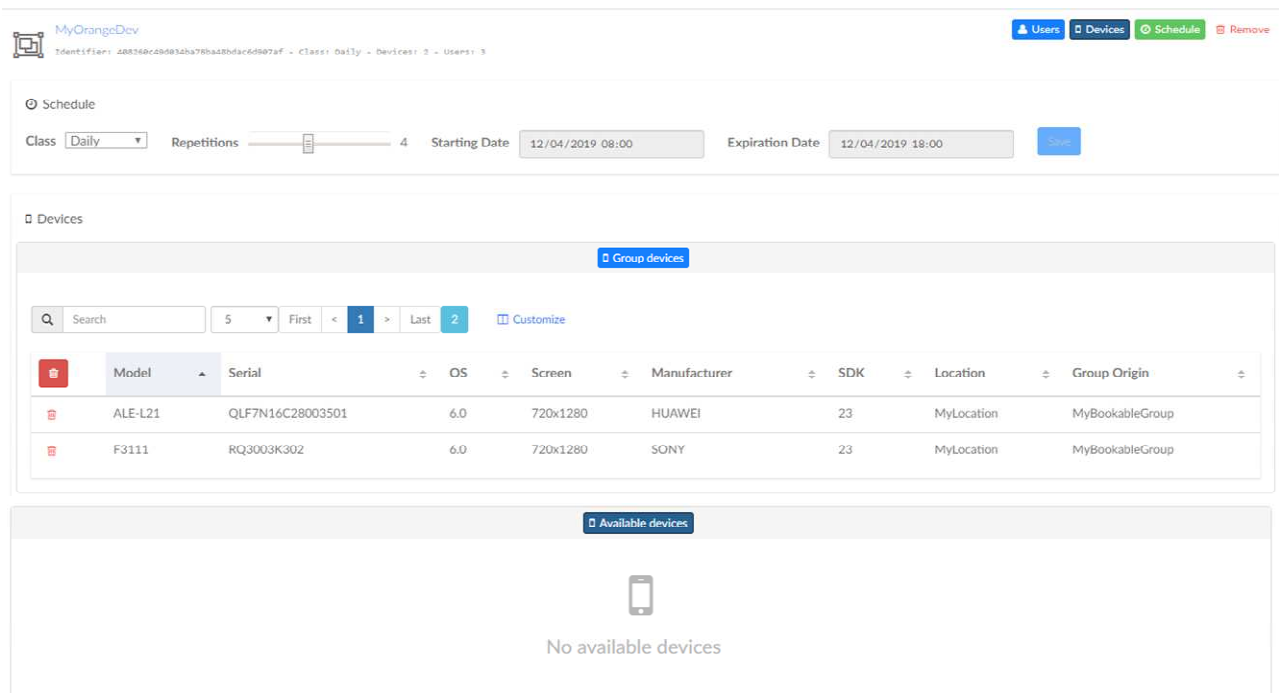
So, she decided to reserve the ALE-21 and F311 models on this period during her working hours (i.e. 8 am – 8 pm), thereby freeing up the considered devices for automated tests triggered outside her working hours.

During this period **lea** will also work with another developer named **bob** who will assist her on an ad hoc basis, involving to include him as a guest user into the device reservation.

So, selecting the **Settings->Groups** tab, we have to create and modify a **transcient** group named **MyOrangeDev**:

-  : create a new group,
-  **Name** : modify its name to **MyOrangeDev** and save it,
-  **Schedule** : modify its schedule as follow and save it
 - class: **Daily**
 - Repetitions: **4**
 - Starting Date: *the day chosen* at **8 am**
 - Expiration Date: *the day chosen* at **8 pm**
-  **Devices** : add **ALE-21** and **F3111** models
-  **Users** : add **lea** and **bob**

Once it is done, we have to get ready the group by clicking on the  **Get ready** button. The Figure 12 below shows the result of the previous actions.



The screenshot displays the configuration for a group named 'MyOrangeDev'. At the top, there are navigation buttons for 'Users', 'Devices', 'Schedule', and 'Remove'. The 'Schedule' section is active, showing a 'Class' dropdown set to 'Daily', 'Repetitions' set to 4, 'Starting Date' as 12/04/2019 08:00, and 'Expiration Date' as 12/04/2019 18:00. Below this, the 'Devices' section is expanded, showing a 'Group devices' table with the following data:

Model	Serial	OS	Screen	Manufacturer	SDK	Location	Group Origin
ALE-L21	QLF7N16C28003501	6.0	720x1280	HUAWEI	23	MyLocation	MyBookableGroup
F3111	RQ3003K302	6.0	720x1280	SONY	23	MyLocation	MyBookableGroup

Below the table, the 'Available devices' section shows a mobile phone icon and the text 'No available devices'.

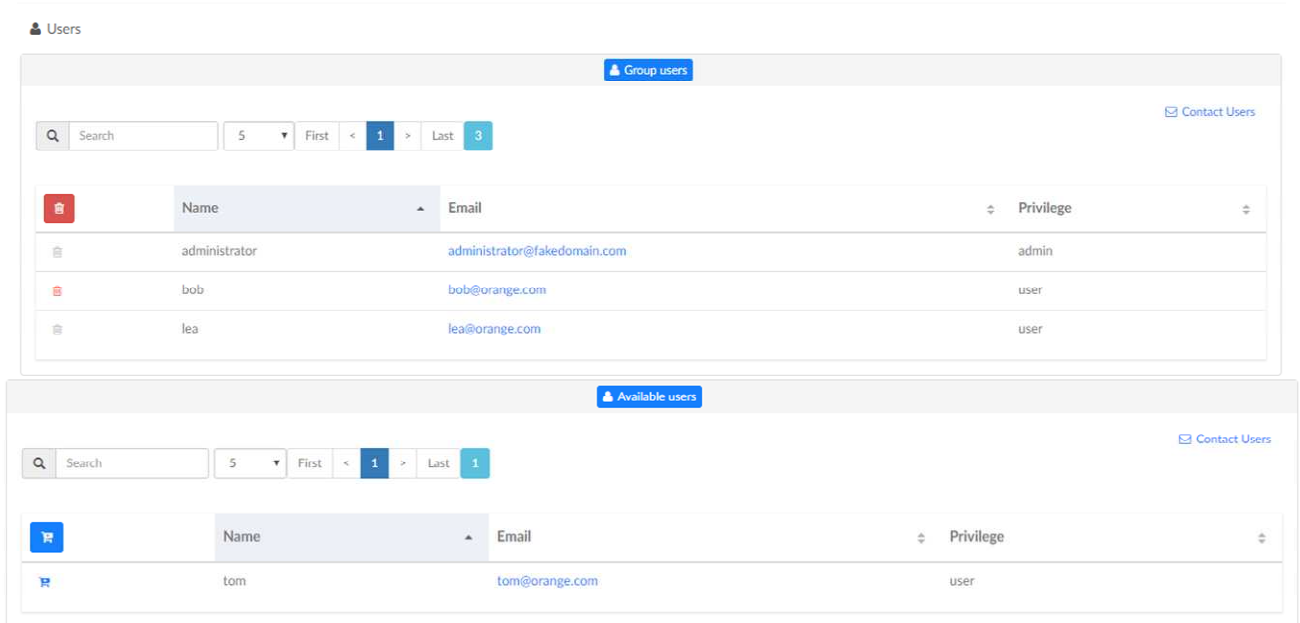


Figure 12: creation of a transient group

Note that selecting the **Settings->Groups** tab, each user can see only the groups for which he is the owner, except the administrator user who see all groups of all users with owner rights.

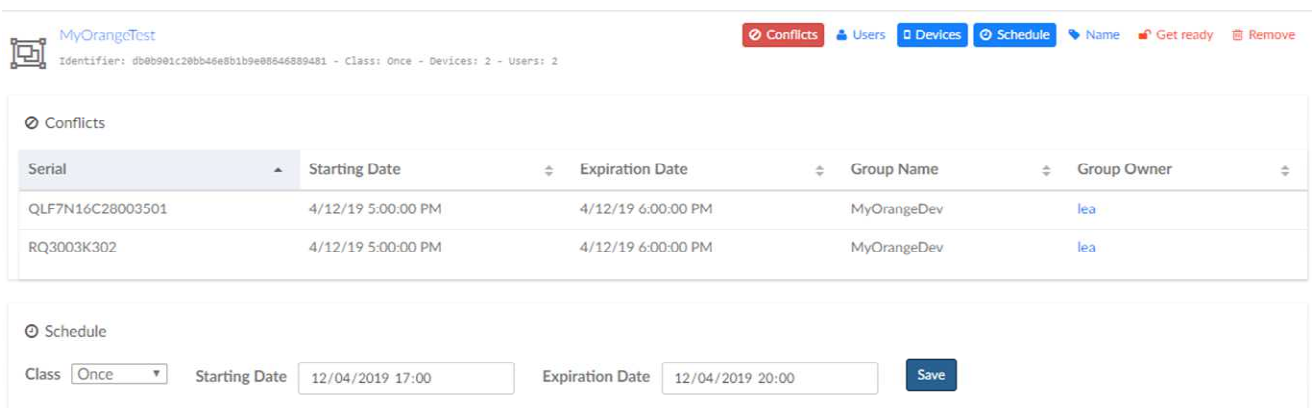
5.4 Facing conflicts

Now, let's imagine **tom** wants to make a reservation (i.e. named **MyOrangeTest**) of the same devices as those reserved by **lea** in **MyOrangeDev** group.

For that, **tom** has to select a group schedule which is not in conflict with the one of **lea**'s group, otherwise the considered devices can't be displayed for adding in the **Available devices** window.

So, once it is done and **tom** has added the devices into his group, let's imagine now **tom** wants to update the group schedule so that it conflicts with the one of **lea**'s group, what happens then?

In such situation, by clicking on the save button of the group schedule, the operation is of course not committed while a list of conflict information is displayed on the UI as shown in the Figure 13 below.



Model	Serial	OS	Screen	Manufacturer	SDK	Location	Group Origin
ALE-L21	QLF7N16C28003501	6.0	720x1280	HUAWEI	23	MyLocation	MyBookableGroup
F3111	RQ3003K302	6.0	720x1280	SONY	23	MyLocation	MyBookableGroup

Figure 13 : Conflicts information

Each conflict per line is detailed as follow:

- the device in conflict
- the timeslot in conflict
- the group name and group owner in conflict

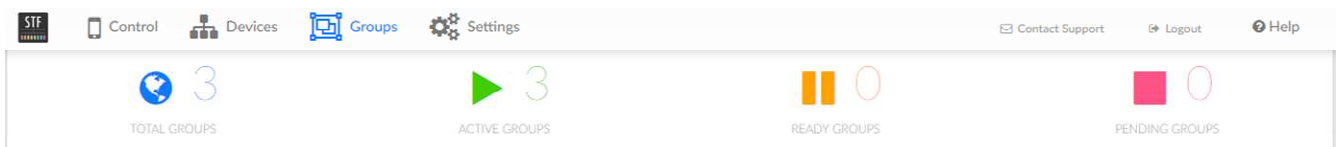
By clicking on the  Conflicts button, the conflict window disappears.

5.5 Monitoring groups

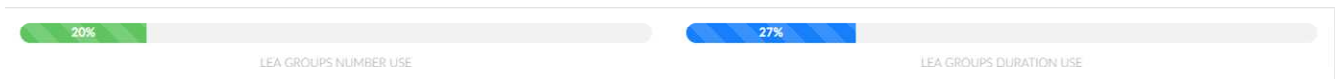
Selecting the **Groups** tab, each user gets a dynamic view of all groups to which he belongs either as an owner or as a guest, as well as other precious statistic information.

In synthesis, this view provides the following information in this order:

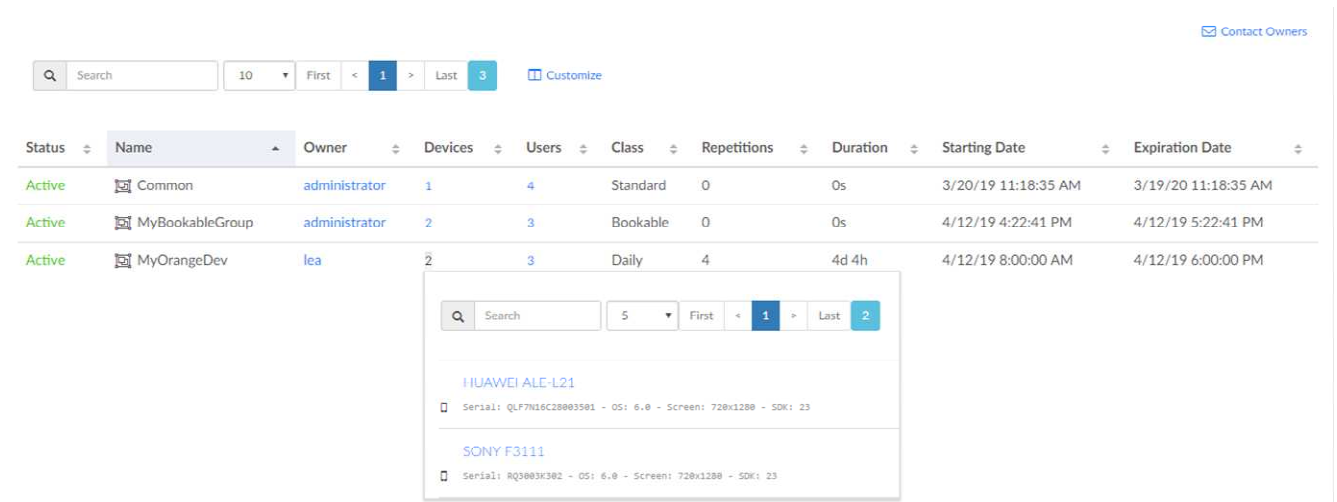
- total number of groups, and number of groups in each state: **pending, ready and active**



- consumption in term of percentage of created groups and percentage of use time of devices for all groups, in relation to the corresponding allocated quotas (cf. section 5.8)



- the list of groups to which the user belongs either as an owner or as a guest, including detailed information for each group (e.g. users, devices, ...)



Note the administrator user view is more complete than the one of a simple user, as illustrated in below Figure 14.

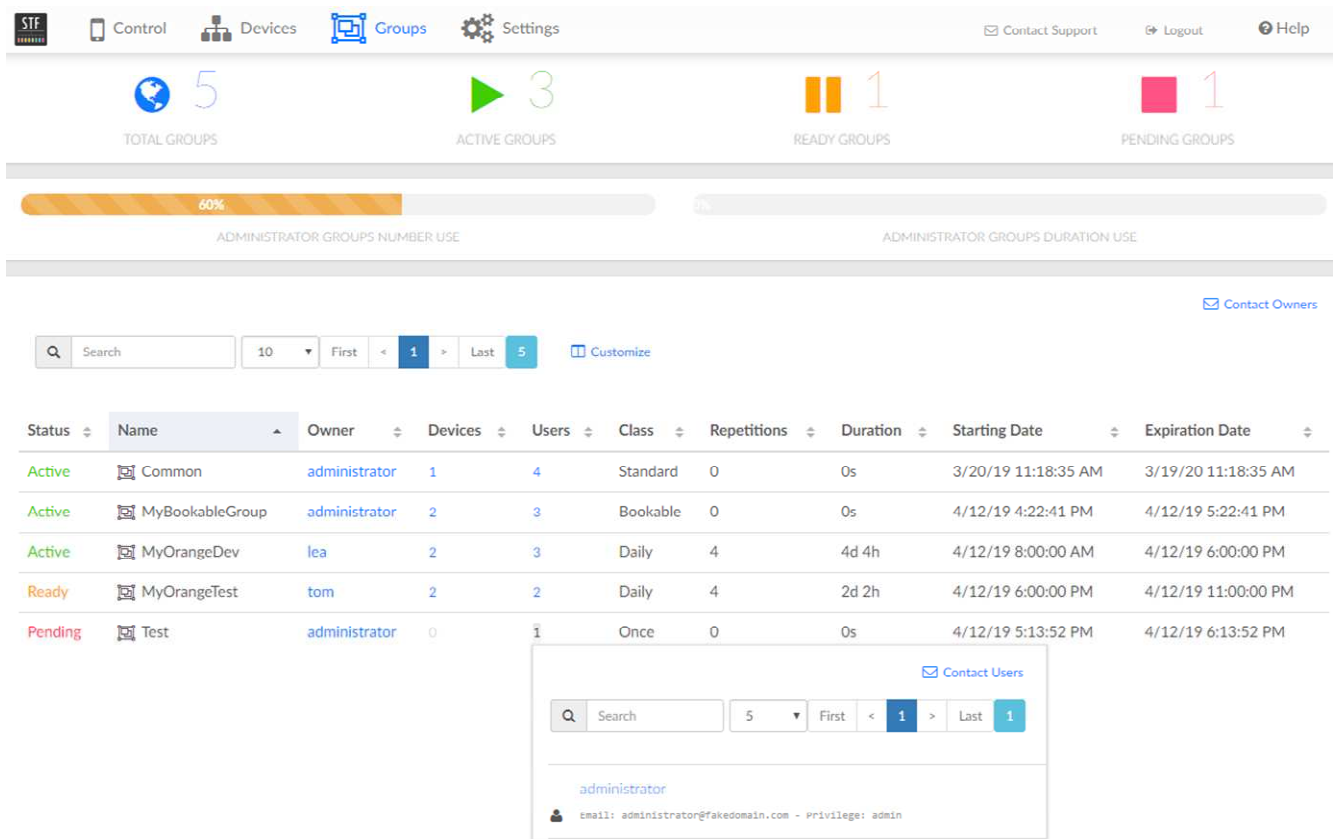


Figure 14: administrator user monitoring groups

In fact, as it can be seen above, the administrator user has a complete view of all groups created by all users!

As a user, it is also possible to contact the group owners or the users of a particular group:

- [Contact Owners](#) : open the default mailbox on the local computer, in order to send an email to a selection of group owners deduced from the search field. The user has then to paste the email addresses from the clipboard

- **Contact Users**: open the default mailbox on the local computer in order to send and email to a selection of users (i.e. search field). The user has then to paste the email addresses from the clipboard.

Note the **Duration** column represents the use time left of devices which is decremented each time a group repetition is done, until the group termination.

The group **Duration** is calculated as follow over time: **(Expiration Date – Starting Date) * (Repetitions + 1) * Devices**

So, by updating the consumed duration quota over time for a group X, it is possible to release some resources in order to add new devices into one or more other groups, without waiting for the effective termination of the considered group X.

5.6 Monitoring devices

Selecting the **Devices** tab, it is possible to customize the view by adding columns relating to the Group feature, as shown in the Figure 15 below.

Lea view

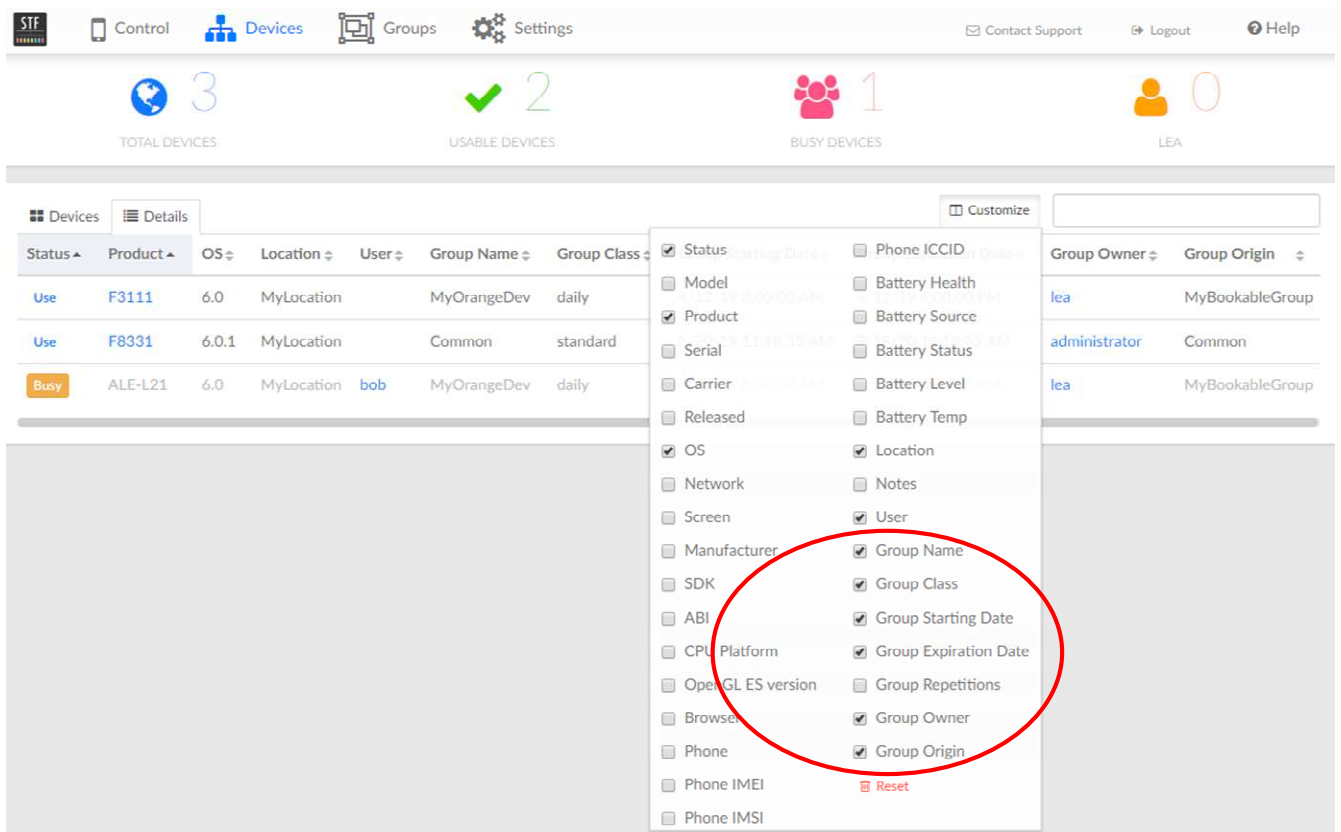


Figure 15: add group columns to lea Devices view

By default, group columns are not enabled in order to keep the default behaviour of STF, so don't hesitate to customize them to fit your preferences!

Note the **Group Name** column corresponds to the name of the current group of the device.

The above **lea** view illustrated by the Figure 15 shows that **bob** is currently using the ALE-21 model during an active state of **MyOrangeDev** group.

It shows also that **lea** may access to all devices because she belongs to both **Common** and **MyBookableGroup** groups.

But what about the view of **bob**, **tom** and administrator user? To find out, let's take a look at the corresponding views !

Tom view

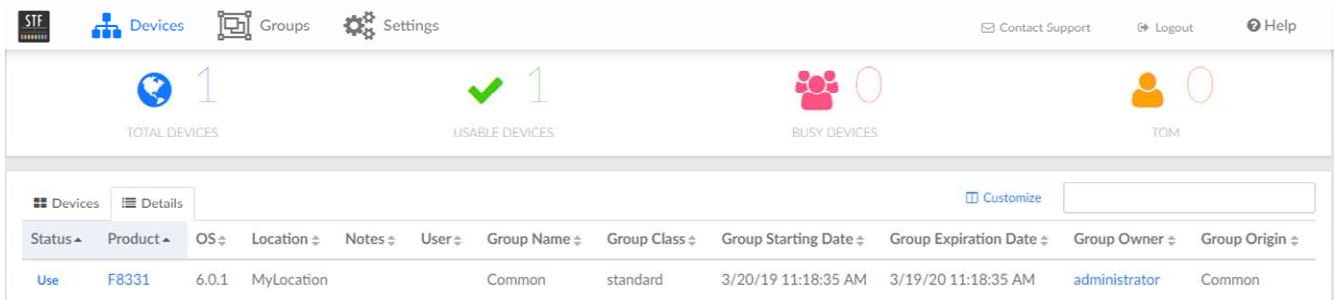


Figure 16: tom monitoring devices

The above Figure 16 shows that **tom** may only access to the F8331 model, because at this time ALE-21 and F3111 models belongs to the active **MyOrangeDev** group in which **tom** is not a member.

Bob view

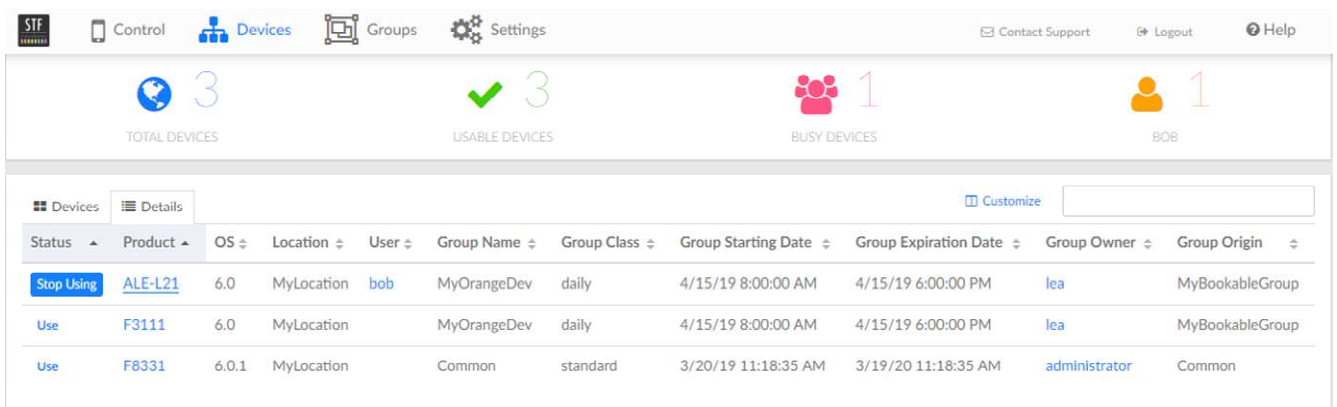


Figure 17: bob monitoring devices

The above Figure 17 shows that bob controls the ALE-21 model and may also access to the F3111 one, because at this time these two devices belongs to the active **MyOrangeDev** group in which **bob** is a member.

Administrator user view

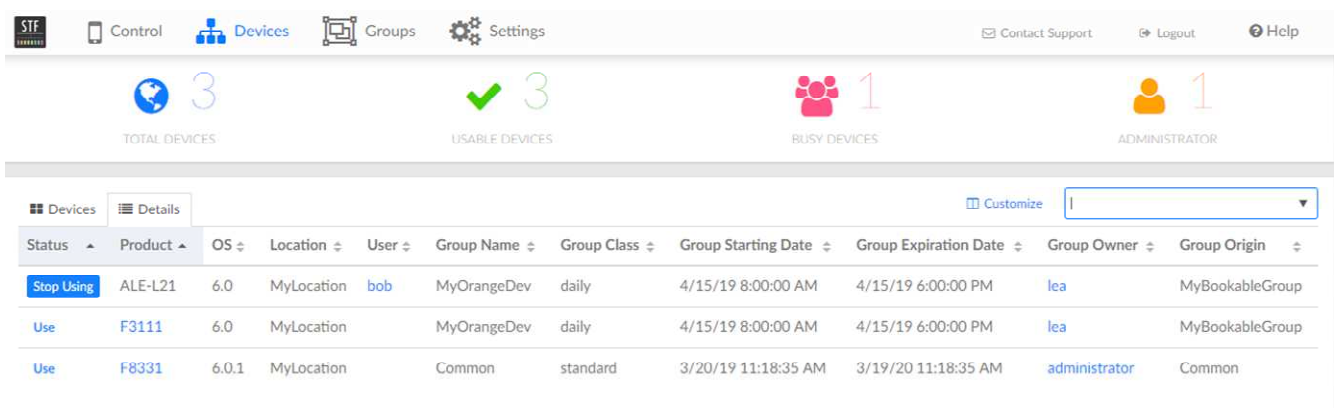


Figure 18: administrator user monitoring devices

The above Figure 18 shows that the administrator user may access to all devices because at any time this user belongs to all groups.

Moreover, the administrator user has also the right to release the device (e.g. ALE-21 model) controlled by any user (e.g. **bob**), but on the other hand he has not the ability to control the device while it is controlled by another user (e.g. **bob**).

5.7 Managing groups

Selecting the **Settings->Groups** tab, it is possible to perform the following actions:

- create & update a group
- remove a selection of groups
- **[administrator user only]** send an email to a selection of group owners

The Figure 19 below shows the general view displayed to the administrator user while managing groups.

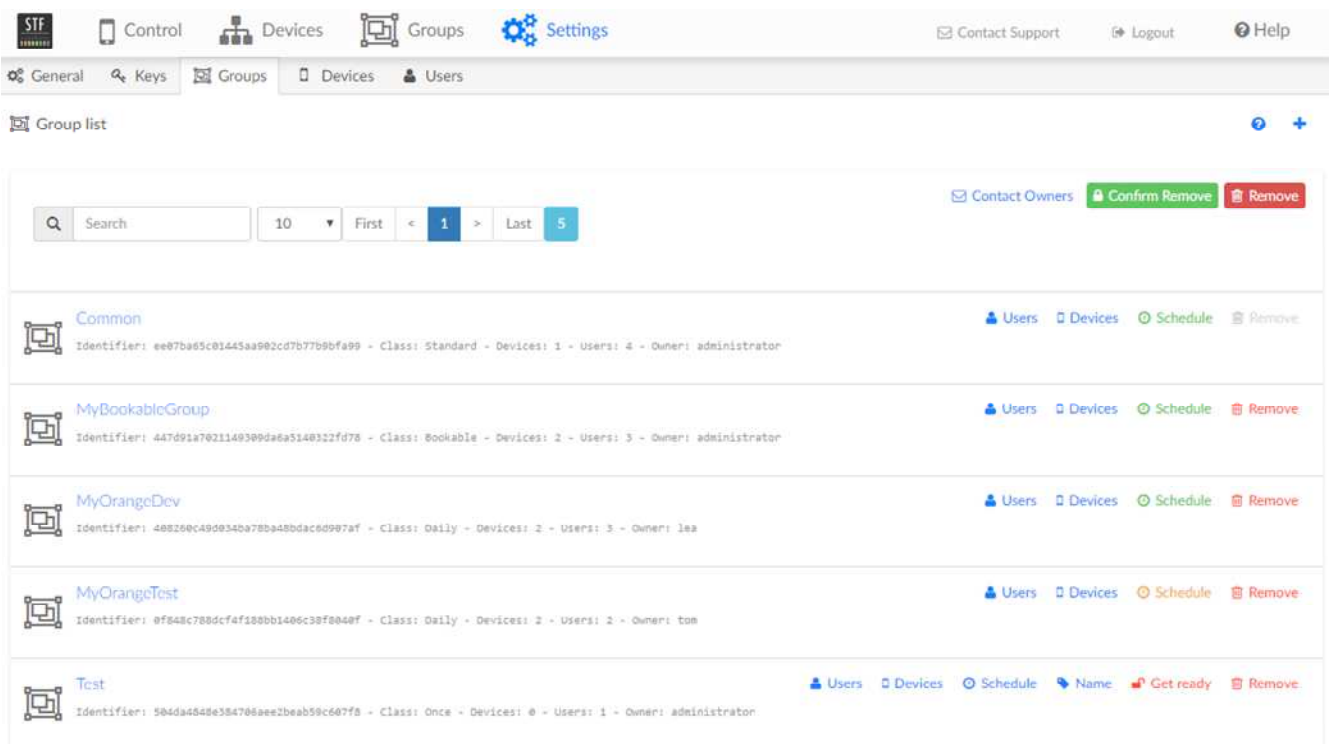











Figure 19: managing groups by the administrator user

- **+** : create a new group
- **Name** : modify the group name
- **Schedule** : modify the group schedule (i.e. **pending** state)
- **Schedule** : display the group schedule (i.e. **ready** state)

-  **Schedule** : display the group schedule (i.e. **active** state)
-  **Devices** : modify the group devices
-  **Users** : modify the group users
-  **Get ready** : get ready the group, meaning the group changes from **pending** state to **ready** state
-  **Remove** : remove a selection of groups (i.e. search field)
-  **Remove** : remove the group
-  **Confirm Remove** : disable the displaying of a confirmation window before removing group(s)
-  **Confirm Remove** : enable the displaying of a confirmation window before removing group(s)
-  **Contact Owners** : open the default mailbox on the local computer, in order to send an email to a selection of group owners deduced from the search field. The user has then to paste the email addresses from the clipboard

Please report to sections 5.2 & 5.3 for group edition details.

On the other hand, the below Figure 20 shows the general view displayed to a simple user (e.g. **lea**) while managing groups.

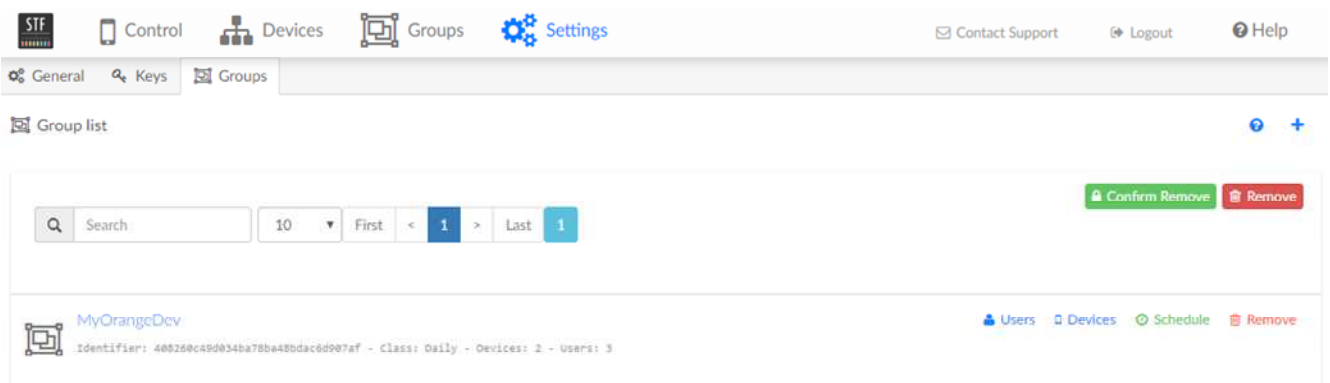


Figure 20: managing groups by a simple user

Note the possible actions are exactly the same than in the administrator user case, except that as a simple user like **lea**, only the groups owned by the user can be obviously displayed, and this is why the implicit group *Owner* property is not displayed either, no more than the *Contact Owners* button which is not relevant in this case.

5.8 Managing users

Privileged feature: you have to login with administrator credentials to perform such operation.

Selecting the **Settings->Users** tab, it is possible to perform the following actions:

- create a user

- remove a filtered selection of users
- update the default groups quotas allocated to each user at its creation time
- update the groups quotas allocated to an individual user after its creation
- send an email to a selection of users

Note that the possibility to create a user before his first login is very useful because it allows defining in advance his rights in term of quotas and group membership!

The Figure 21 below shows the general view while managing users.

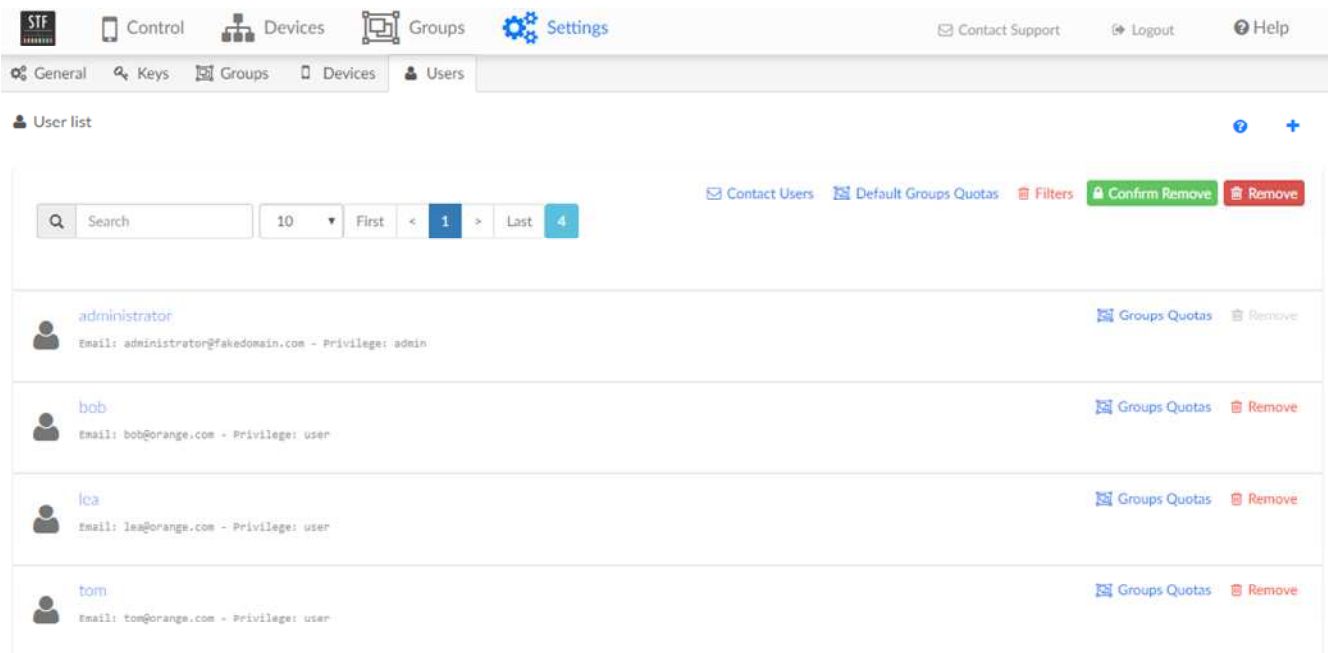
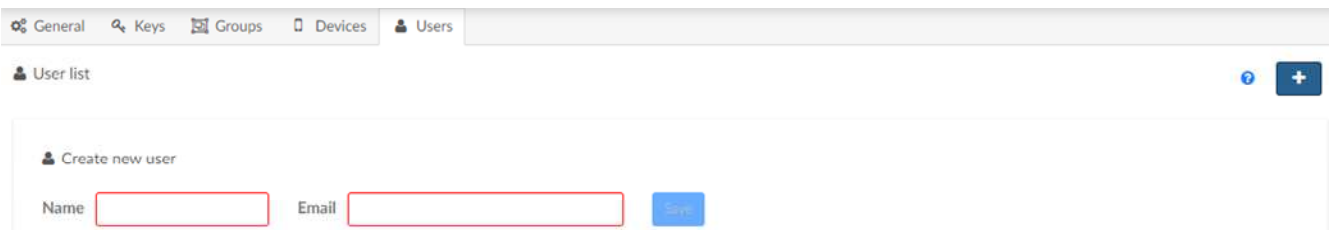


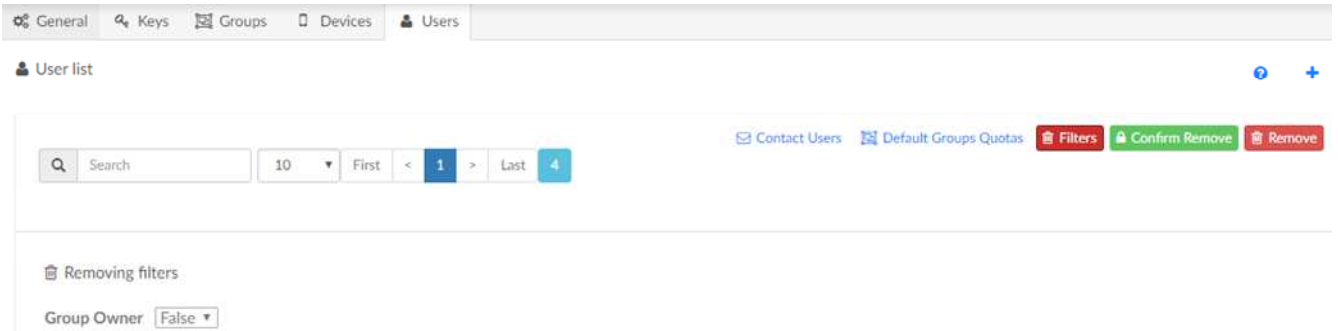
Figure 21: managing users

- **+** : create a new user





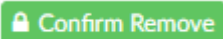


Note: the **Name** field must conform to this regular expression : `/^[0-9a-zA-Z_\.]{1,50}$/`

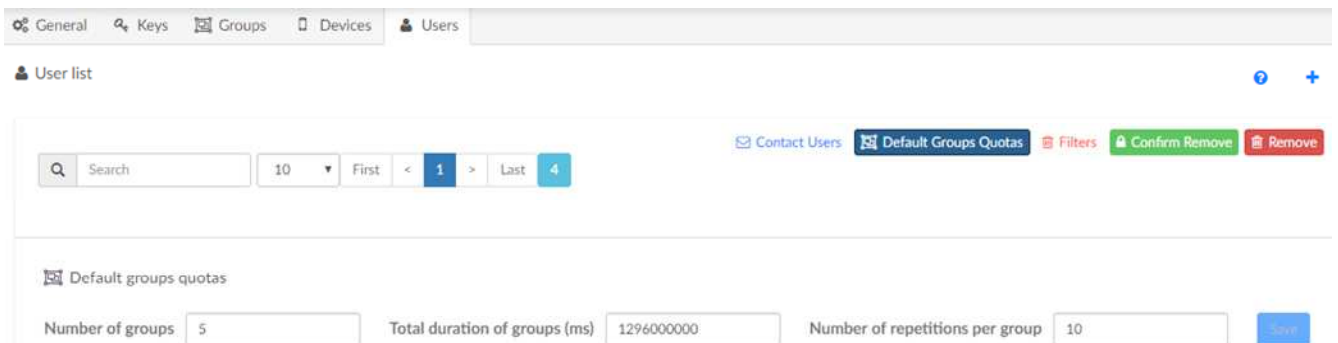
- **Filters** : set a filter for removing user(s) action applying on the selection of users



The filter may have one the following values:

- **False:** the user is not a group owner (i.e. default value)
- **True:** the user is a group owner
- **Any:** no filter

-  : remove a filtered selection of users (i.e. search field)
-  : remove the filtered user
-  : disable the displaying of a confirmation window before removing user(s)
-  : enable the displaying of a confirmation window before removing user(s)
-  : update the default groups quotas allocated to each user at its creation time



Notes:

- **Number of groups:** maximum number of groups that a user is allowed to create; default value = 5
- **Total duration of groups (ms):** duration in milliseconds which represents the maximum use time of devices; default value = 1296000000 (i.e. ~ 15 day 1 device ~ 5 days 3 devices etc.)

This global use time of devices if the sum of the use times of devices allocated to each group created by the user, knowing that each use time of devices per group, named also group duration, is calculated as follow over time:

$$\triangleright (\text{Expiration Date} - \text{Starting Date}) * (\text{Repetitions} + 1) * \text{Devices}$$

So, by updating the consumed duration quota over time that is at the end of each repetition it is possible to release some resources in order to add new devices in one or more groups, without waiting for the effective termination of the considered groups.

- **Number of repetitions per group:** maximum number of repetitions a group created by a user may have; default value = 10

User quotas are designed to avoid some abnormal behaviors, such as:

- reservation of all devices by a single user
- practically infinite group duration
- ...

On the other hand, if a user meets some limitations with the default allocated quotas, he has just to ask to the administrator user to adjust his quotas to fit his needs as described hereafter.

- [Groups Quotas](#) : update the groups quotas allocated to the user after its creation

The screenshot shows the user profile for 'lea' (Email: lea@orange.com - Privilege: user) with a 'Groups Quotas' button and a 'Remove' button. Below this, the 'Groups Quotas' section contains three input fields: 'Number of groups' with the value 7, 'Total duration of groups (ms)' with the value 1296000000, and 'Number of repetitions per group' with the value 15. A 'Save' button is located to the right of these fields.

The fields have the same definition as for **Default Groups Quotas**, the difference is that **Groups Quotas** updating affects the corresponding user rights in real time.

However, note it is not possible to reduce a user quota if the current consumed quota is greater than the proposed allocated quota.

For instance, if **lea** has yet created 7 groups, it is not possible to change the corresponding allocated quota value to 6; you have to wait for at least a group termination to do that.

- [Contact Users](#) : open the default mailbox on the local computer in order to send and email to a selection of users (i.e. search field). The user has then to paste the email addresses from the clipboard.

5.9 Managing devices

Privileged feature: you have to login with administrator credentials to perform such operation.

Selecting the Settings->Devices tab, it is possible to perform the following action:

- remove a filtered selection of devices

The Figure 22 below shows the general view while managing devices.

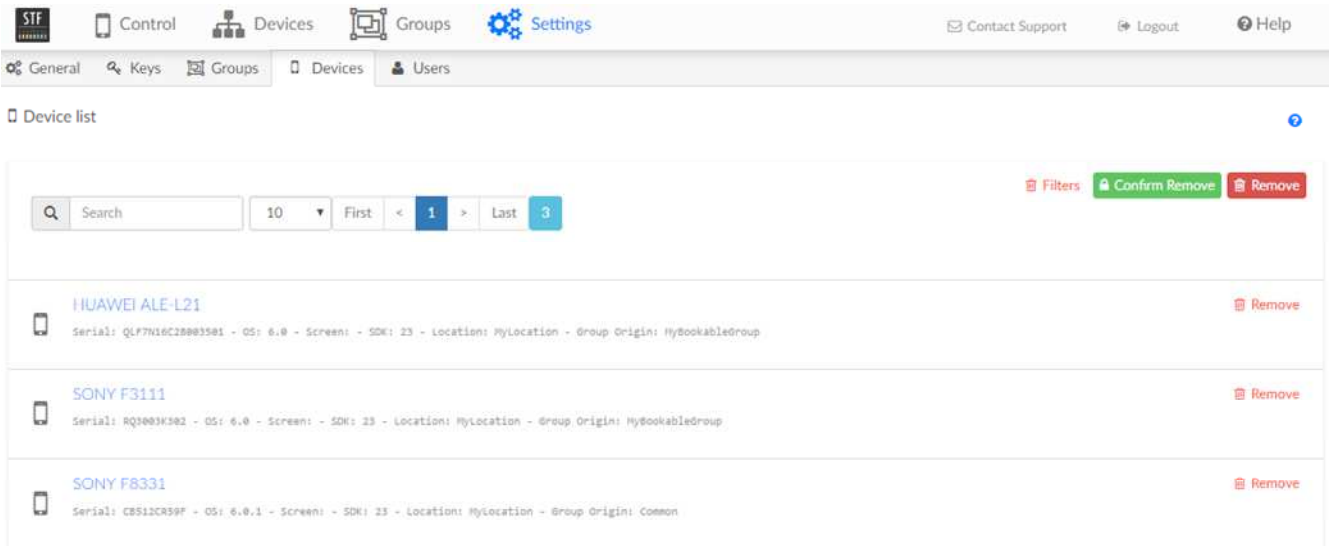
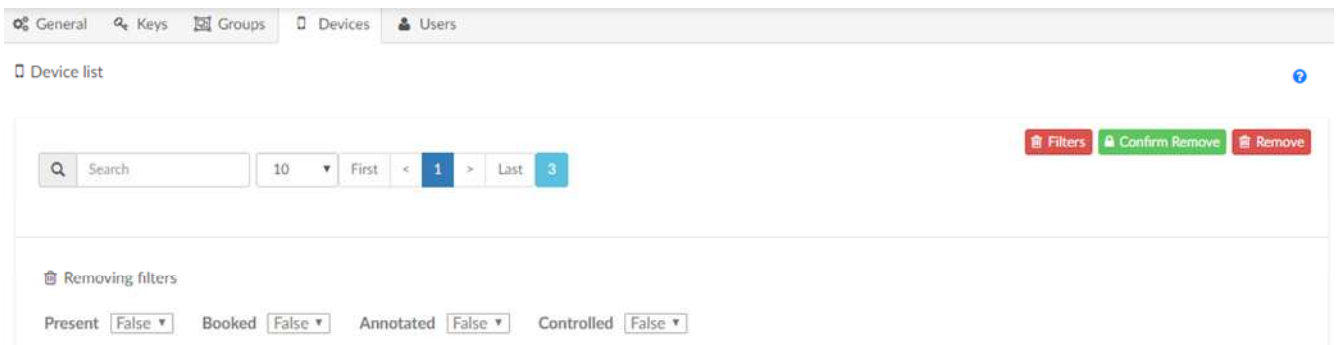


Figure 22: managing devices

- **Filters** : set filters for removing device(s) action applying on the selection of devices



The **Present** filter may have one the following values:

- **False**: the device is not present (i.e. default value)
- **True**: the device is present
- **Any**: no filter

The **Booked** filter may have one the following values:





- **False**: the device is not booked (i.e. default value)
- **True**: the device is booked
- **Any**: no filter

The **Annotated** filter may have one the following values:

- **False**: the device **Notes** field is not filled (i.e. default value)
- **True**: the device **Notes** field is filled
- **Any**: no filter

The **Controlled** filter may have one the following values:

- **False**: the device is not controlled by a user (i.e. default value)
- **True**: the device is controlled by a user
- **Any**: no filter

-  **Remove** : remove a filtered selection of devices (i.e. search field)
-  **Remove** : remove the filtered device
-  **Confirm Remove** : disable the displaying of a confirmation window before removing device(s)
-  **Confirm Remove** : enable the displaying of a confirmation window before removing device(s)

6 RESTful API (add-on)

Due to the introduction of the Group concept along with an administrator level, the STF API has undergone many changes while maintaining backward compatibility with client side software:

- Some legacy APIs have undergone an implementation change, without impact on their semantic and syntax
 - e.g. the API `GET /user/device/{serial}` implementation has been modified in order to verify that the device is part of the device universe of the user (i.e. access right to the device), otherwise a `Device not found` error message is returned
- In addition of an implementation change, some other legacy APIs have undergone a syntactic enrichment, preserving their original syntax and without impact on their original semantic
 - e.g. the API `GET /devices` syntax has been enriched with an optional query parameter whose default value is to keep the legacy behaviour
- Finally, some new APIs have been introduced
 - e.g. API `GET /groups`

So, the next sections will describe only the new APIs (i.e. tagged as **New**), as well as those whose syntax has been enriched (i.e. tagged as **Enriched**).

Note due to the introduction of an administrator level in addition of the legacy user one, some APIs will be tagged as **Privileged** meaning only the administrator user can use them.

For more details about the complete STF API, please report to the corresponding swagger file `api_v1.yaml` [7] or report to the section 8 « Using Swagger UI »

6.1 User management

6.1.1 Access tokens

New

GET /user/fullAccessTokens

Returns your access tokens; note that all access token fields are returned in response including the **id** one.

Using cURL:

```
curl -X GET -H "authorization: Bearer YOUR-TOKEN" http://@IP/api/v1/user/fullAccessTokens
```

New

GET /user/accessTokens/{id}

Returns one of your access tokens.

Using cURL:

```
curl -X GET -H "authorization: Bearer YOUR-TOKEN" http://@IP/api/v1/user/accessTokens/{id}
```


New

POST /user/accessTokens?title={string}

Creates an access token for you.

Using cURL:

```
curl -X POST -H "authorization: Bearer YOUR-TOKEN" http://@IP/api/v1/user/accessTokens?title={string}
```

New

DELETE /user/accessToken/{id}

Removes one of your access tokens.

Using cURL:

```
curl -X DELETE -H "authorization: Bearer YOUR-TOKEN" http://@IP/api/v1/user/accessTokens/{id}
```

New

DELETE /user/accessTokens

Removes your access tokens.

Using cURL:

```
curl -X DELETE -H "authorization: Bearer YOUR-TOKEN" http://@IP/api/v1/user/accessTokens
```

6.1.2 Devices

POST /user/devices/{serial}?timeout={integer}

Places a device under user control; note this is not completely analogous to press the 'Use' button in the UI because that does not authorize remote connection through ADB.

The optional **timeout** query parameter must be positive; it means if the device is kept idle for this period (in milliseconds), it will be automatically removed from the user control. Default value is provided by the provider group timeout parameter.

Using cURL:

```
curl -X POST -H "authorization: Bearer YOUR-TOKEN" http://@IP/api/v1/user/devices/{serial}?timeout={integer}
```

6.2 Users management

6.2.1 Users

New

GET /users?fields={string}

Returns the users; if you are the administrator user then all user fields are returned, otherwise only **email**, **name** and **privilege** user fields are returned.

Using cURL:

```
curl -X GET -H "authorization: Bearer YOUR-TOKEN" http://@IP/api/v1/users?fields={string}
```

The optional **fields** query parameter takes a comma-separated list of user fields; only listed fields will be returned in response.

New

GET /users/{email}?fields={string}

Returns a user; if you are the administrator user then all user fields are returned, otherwise only **email**, **name** and **privilege** user fields are returned.

Using cURL:

```
curl -X GET -H "authorization: Bearer YOUR-TOKEN" http://@IP/api/v1/users/{email}?fields={string}
```

The optional **fields** query parameter takes a comma-separated list of user fields; only listed fields will be returned in response.

Privileged New

POST /users/{email}?name={string}

Creates a user in the database.

Using cURL:

```
curl -X POST -H "authorization: Bearer YOUR-TOKEN" http://@IP/api/v1/users/{email}?name={string}
```

Privileged New

DELETE /users/{email}?groupOwner={boolean}

Removes a user from the database.

The optional **groupOwner** query parameter allows or not the removing of the user depending respectively if the user is a group owner (**true**) or not (**false**); note that by not providing the **groupOwner** parameter it means an unconditionally removing.

Using cURL:

```
curl -X DELETE -H "authorization: Bearer YOUR-TOKEN" http://@IP/api/v1/users/{email}?groupOwner={boolean}
```

Privileged New

DELETE /users?groupOwner={boolean}

Removes users from the database.

The optional **groupOwner** query parameter allows or not the removing of each user depending respectively if the user is a group owner (**true**) or not (**false**); note that by not providing the **groupOwner** parameter it means an unconditionally removing.

The optional **users** body parameter specifies users to remove as a comma-separated list of emails; note that by not providing this parameter it means all users are selected for removing.

Using cURL:

```
curl -X DELETE --header "Content-Type: application/json" --data '{"users": "EMAILS"}' -H "authorization: Bearer YOUR-TOKEN" http://@IP/api/v1/users?groupOwner={boolean}
```

6.2.2 Groups quotas

Privileged **New**

PUT /users/groupsQuotas?number={integer}&duration={integer}&repetitions={integer}

Updates the default groups quotas allocated to each new user.

Note that all query parameters are optional and must be positive.

Using cURL:

```
curl -X PUT -H "authorization: Bearer YOUR-TOKEN" http://@IP/api/v1/users/groupsQuotas?number={integer}&duration={integer}&repetitions={integer}
```

Privileged **New**

PUT /users/{email}/groupsQuotas?number={integer}&duration={integer}&repetitions={integer}

Updates the groups quotas of a user.

Note that all query parameters are optional and must be positive.

Using cURL:

```
curl -X PUT -H "authorization: Bearer YOUR-TOKEN" http://@IP/api/v1/users/{email}/groupsQuotas?number={integer}&duration={integer}&repetitions={integer}
```

6.2.3 Access tokens

Privileged **New**

GET /users/{email}/accessTokens/{id}

Returns an access token of a user.

Using cURL:

```
curl -X GET -H "authorization: Bearer YOUR-TOKEN" http://@IP/api/v1/users/{email}/accessTokens/{id}
```

Privileged **New**

GET /users/{email}/accessTokens

Returns the access tokens of a user.

Using cURL:

```
curl -X GET -H "authorization: Bearer YOUR-TOKEN" http://@IP/api/v1/users/{email}/accessTokens
```

Privileged **New**

POST /users/{email}/accessTokens?title={string}

Creates an access token for a user.

Using cURL:

```
curl -X POST -H "authorization: Bearer YOUR-TOKEN" http://@IP/api/v1/users/{email}/accessTokens?title={string}
```

Privileged **New**

DELETE /users/{email}/accessTokens/{id}

Removes an access token of a user.

Using cURL:

```
curl -X DELETE -H "authorization: Bearer YOUR-TOKEN" http://@IP/api/v1/users/{email}/accessTokens/{id}
```

Privileged **New**

DELETE /users/{email}/accessTokens

Removes the access tokens of a user.

Using cURL:

```
curl -X DELETE -H "authorization: Bearer YOUR-TOKEN" http://@IP/api/v1/users/{email}/accessTokens
```

6.2.4 Devices

Privileged **New**

GET /users/{email}/devices/{serial}?fields={string}

Returns a device controlled by a user.

Using cURL:

```
curl -X GET -H "authorization: Bearer YOUR-TOKEN" http://@IP/api/v1/users/{email}/devices/{serial}
```

The optional **fields** query parameter takes a comma-separated list of device fields; only listed fields will be returned in response.

Privileged **New**

GET /users/{email}/devices?fields={string}

Returns the devices controlled by a user.

Using cURL:

```
curl -X GET -H "authorization: Bearer YOUR-TOKEN" http://@IP/api/v1/users/{email}/devices?fields={string}
```

The optional **fields** query parameter takes a comma-separated list of device fields; only listed fields will be returned in response.

Privileged **New**

POST /users/{email}/devices/{serial}?timeout={integer}

Places a device under user control; note this is not completely analogous to press the 'Use' button in the UI because that does not authorize remote connection through ADB.

The optional **timeout** query parameter must be positive; it means if the device is kept idle for this period (in milliseconds), it will be automatically removed from the user control. Default value is provided by the provider group timeout parameter.

Using cURL:

```
curl -X POST -H "authorization: Bearer YOUR-TOKEN" http://@IP/api/v1/users/{email}/devices/{serial}?timeout={integer}
```

Privileged **New**

DELETE /users/{email}/devices/{serial}

Remove a device from the user control; note this is analogous to press the 'Stop Using' button in the UI because that forbids also remote connection through ADB.

Using cURL:

```
curl -X DELETE -H "authorization: Bearer YOUR-TOKEN" http://@IP/api/v1/users/{email}/devices/{serial}
```

Privileged **New**

POST /users/{email}/devices/{serial}/remoteConnect

Allows to remotely connect to a device controlled by a user; returns the remote debug URL in response for use with ADB.

Using cURL:

```
curl -X POST -H "authorization: Bearer YOUR-TOKEN" http://@IP/api/v1/users/{email}/devices/{serial}/remoteConnect
```

Privileged **New**

DELETE /users/{email}/devices/{serial}/remoteConnect

Forbids using ADB to remotely connect to a device controlled by a user.

Using cURL:

```
curl -X DELETE -H "authorization: Bearer YOUR-TOKEN" http://@IP/api/v1/users/{email}/devices/{serial}/remoteConnect
```

6.3 Devices management

6.3.1 Devices

Enriched

GET /devices?target={string}&fields={string}

Returns the devices of your universe.

Note the query **fields** parameter is optional (i.e. comma-separated list of device fields).

The query **target** parameter is also optional; the string must conform to the following enumerator:

- **bookable**
 - devices belonging to a bookable group
- **standard**
 - devices belonging to a standard group
- **origin**
 - all devices
- **standardizable**
 - targets devices which are not yet booked including those belonging to a standard group
- **user** (i.e. default value)
 - targets devices which are accessible by you at a given time

Using cURL:

```
curl -X GET -H "authorization: Bearer YOUR-TOKEN" http://@IP/api/v1/devices?target={string}&fields={string}
```

Privileged **New**

DELETE /devices?present={boolean}&booked={boolean}&annotated={boolean}&controlled={boolean}

Removes devices from the database.

The optional **present** query parameter allows or not the removing of each device depending respectively if the device is present (**true**) or not (**false**); note that by not providing this parameter it means an unconditional removing.

The optional **booked** query parameter allows or not the removing of each device depending respectively if the device is booked (**true**) or not (**false**); note that by not providing this parameter it means an unconditional removing.

The optional **annotated** query parameter allows or not the removing of each device depending respectively if the device is annotated (**true**) or not (**false**); note that by not providing this parameter it means an unconditional removing.

The optional **controlled** query parameter allows or not the removing of each device depending respectively if the device is controlled (**true**) or not (**false**); note that by not providing this parameter it means an unconditional removing.

The optional **devices** body parameter specifies devices to remove as a comma-separated list of serials; note that by not providing this parameter it means all devices are selected for removing.

Using cURL:

```
curl -X DELETE --header "Content-Type: application/json" --data '{"devices": "SERIALS"}' -H "authorization: Bearer YOUR-TOKEN" http://@IP/api/v1/users?present={boolean}&booked={boolean}&annotated={boolean}&controlled={boolean}
```

Privileged New

DELETE /devices/{serial}?present={boolean}&booked={boolean}&annotated={boolean}&controlled={boolean}

Removes a device from the database.

The optional **present** query parameter allows or not the removing of the device depending respectively if the device is present (**true**) or not (**false**); note that by not providing this parameter it means an unconditional removing.

The optional **booked** query parameter allows or not the removing of the device depending respectively if the device is booked (**true**) or not (**false**); note that by not providing this parameter it means an unconditional removing.

The optional **annotated** query parameter allows or not the removing of the device depending respectively if the device is annotated (**true**) or not (**false**); note that by not providing this parameter it means an unconditional removing.

The optional **controlled** query parameter allows or not the removing of the device depending respectively if the device is controlled (**true**) or not (**false**); note that by not providing this parameter it means an unconditional removing.

Using cURL:

```
curl -X DELETE -H "authorization: Bearer YOUR-TOKEN" http://@IP/api/v1/users?present={boolean}&booked={boolean}&annotated={boolean}&controlled={boolean}
```

6.3.2 Groups

Privileged New

GET /devices/{serial}/groups?fields={string}

Returns all groups to which the device belongs.

Using cURL:

```
curl -X GET -H "authorization: Bearer YOUR-TOKEN" http://@IP/api/v1/devices/{serial}/groups?fields={string}
```

The optional **fields** query parameter takes a comma-separated list of group fields; only listed fields will be returned in response.

New

GET /devices/{serial}/bookings?fields={string}

Returns all bookings (i.e. transient groups) to which the device belongs.

Using cURL:

```
curl -X GET -H "authorization: Bearer YOUR-TOKEN" http://@IP/api/v1/devices/{serial}/bookings?fields={string}
```

The optional **fields** query parameter takes a comma-separated list of group fields; only listed fields will be returned in response.

Privileged New

PUT /devices/groups/{id}?fields={string}

Adds devices into an origin group along with updating each device; returns the updated devices.

The optional **devices** body parameter specifies devices to add as a comma-separated list of serials; note that by not providing this parameter it means all **available devices** are selected for adding:

- **available devices** means all devices in case of a bookable group
- **available devices** means all not yet booked devices in case of a standard group

The optional **fields** query parameter takes a comma-separated list of device fields; only listed fields will be returned in response.

Using cURL:

```
curl -X PUT --header "Content-Type: application/json" --data '{"devices": "SERIALS"}' -H "authorization: Bearer YOUR-TOKEN" http://@IP/api/v1/devices/groups/{id}
```

Privileged New

PUT /devices/{serial}/groups/{id}

Adds a device into the origin group along with updating the added device; returns the updated device.

Using cURL:

```
curl -X PUT -H "authorization: Bearer YOUR-TOKEN" http://@IP/api/v1/devices/{serial}/groups/{id}
```


Privileged **New**

DELETE /devices/groups/{id}?fields={string}

Removes devices from an origin group along with updating each removed device; returns the updated devices.

The optional **devices** body parameter specifies devices to remove as a comma-separated list of serials; note that by not providing this parameter it means all devices of the group are selected for removing.

The optional **fields** query parameter takes a comma-separated list of device fields; only listed fields will be returned in response.

Using cURL:

```
curl -X DELETE --header "Content-Type: application/json" --data '{"devices": "SERIALS"}' -H "authorization: Bearer YOUR-TOKEN" http://@IP/api/v1/devices/groups/{id}
```

Privileged **New**

DELETE /devices/{serial}/groups/{id}

Removes a device from the origin group along with updating the removed device; returns the updated device.

Using cURL:

```
curl -X DELETE -H "authorization: Bearer YOUR-TOKEN" http://@IP/api/v1/devices/{serial}/groups/{id}
```

6.4 Groups management

6.4.1 Groups

New

GET /groups/{id}?fields={string}

Returns a group to which you belong.

Using cURL:

```
curl -X GET -H "authorization: Bearer YOUR-TOKEN" http://@IP/api/v1/groups/{id}?fields={string}
```

The optional **fields** query parameter takes a comma-separated list of group fields; only listed fields will be returned in response.

New

GET /groups?owner={boolean}&fields={string}

Returns the groups to which you belong.

The optional **owner** query parameter selects the groups for which you are the owner (**true**) or a simple member (**false**); note that by not providing this parameter, it means all groups to which you belong are selected.

Using cURL:

```
curl -X GET -H "authorization: Bearer YOUR-TOKEN" http://@IP/api/v1/groups?owner={boolean}&fields={string}
```

The optional **fields** query parameter takes a comma-separated list of group fields; only listed fields will be returned in response.

New

DELETE /groups/{id}

Removes a group owned by you.

Using cURL:

```
curl -X DELETE -H "authorization: Bearer YOUR-TOKEN" http://@IP/api/v1/groups/{id}
```

New

DELETE /groups

Removes the groups owned by you.

The optional **groups** body parameter specifies the groups to remove as a comma-separated list of identifiers; note that by not providing this parameter it means all groups owned by you are removed

Using cURL:

```
curl -X DELETE -H "authorization: Bearer YOUR-TOKEN" --data '{"groups": "IDs"}' http://@IP/api/v1/groups
```

New

PUT /groups/{id}

Updates a group owned by you.

The group's properties are provided as body parameters; at least one property is required.

Using cURL:

```
curl -X PUT -H "authorization: Bearer YOUR-TOKEN" --header "Content-Type: application/json" --data '{"name": "patterned string", "startTime": "date-time string", "stopTime": "date-time string", "class": "enum string", "repetitions": "positive integer", "state": "enum string"}' http://@IP/api/v1/groups/{id}
```

name pattern: /^[0-9a-zA-Z-_.:/]{1,50}\$/

state enumeration: pending, ready

class enumeration: once, bookable, hourly, daily, weekly, monthly, quarterly, halfyearly, yearly, debug, standard

New

POST /groups

Creates a group with you as owner.

The group's properties are provided as body parameters; at least one property is required.

Using cURL:

```
curl -X POST -H "authorization: Bearer YOUR-TOKEN" --header "Content-Type: application/json" --data '{"name": "patterned string", "startTime": "date-time string", "stopTime": "date-time string", "class": "enum string", "repetitions": "positive integer", "state": "enum string"}' http://@IP/api/v1/groups/{id}
```

startTime default value: *creation time*

stopTime default value: **startTime** + 1 hour

repetitions default value: 0

name pattern: `/^[0-9a-zA-Z-_.:/:]{1,50}$/`
- default value: generated at runtime

state enumeration: `pending, ready`
- default value: `pending` or `ready` for bookable/standard classes

class enumeration: `once, bookable, hourly, daily, weekly, monthly, quarterly, halfyearly, yearly, debug, standard`
- default value: `once`
- privileged values: `debug, bookable, standard`

6.4.2 Devices

New

GET /groups/{id}/devices/{serial}?fields={string}

Returns a device of a group to which you belong.

Using cURL:

```
curl -X GET -H "authorization: Bearer YOUR-TOKEN" http://@IP/api/v1/groups/{id}/devices/{serial}?fields={string}
```

The optional **fields** query parameter takes a comma-separated list of device fields; only listed fields will be returned in response.

New

GET /groups/{id}/devices?bookable={boolean}&fields={string}

Returns the devices of the group to which you belong.

The optional **bookable** query parameter selects devices which could be potentially booked by that transient group (**true** -> irrelevant for an origin group!), or selects all devices of the group (**false**); note that by not providing this parameter all devices of the group are selected.

Using cURL:

```
curl -X GET -H "authorization: Bearer YOUR-TOKEN" http://@IP/api/v1/groups/{id}/devices?bookable={boolean}&fields={string}
```

The optional **fields** query parameter takes a comma-separated list of device fields; only listed fields will be returned in response.

New

PUT /groups/{id}/devices/{serial}

Adds a device into a transient group owned by you.

Using cURL:

```
curl -X PUT -H "authorization: Bearer YOUR-TOKEN" http://@IP/api/v1/groups/{id}/devices/{serial}
```

New

PUT /groups/{id}/devices

Adds devices into a transient group owned by you.

The optional **devices** body parameter specifies devices to add as a comma-separated list of serials; note that by not providing this parameter it means all devices which could be potentially booked by that transient group are added into the latter.

Using cURL:

```
curl -X PUT -H "authorization: Bearer YOUR-TOKEN" --data '{"devices": "SERIALS"}' http://@IP/api/v1/groups/{id}/devices
```

New

DELETE /groups/{id}/devices/{serial}

Removes a device from a transient group owned by you.

Using cURL:

```
curl -X DELETE -H "authorization: Bearer YOUR-TOKEN" http://@IP/api/v1/groups/{id}/devices/{serial}
```

New

DELETE /groups/{id}/devices

Removes devices from a transient group owned by you.

The optional **devices** body parameter specifies the devices to remove as a comma-separated list of serials; note that by not providing this parameter it means all devices of the group are removed.

Using cURL:

```
curl -X DELETE -H "authorization: Bearer YOUR-TOKEN" --data '{"devices": "SERIALS"}' http://@IP/api/v1/groups/{id}/devices
```

6.4.3 Users

New

GET /groups/{id}/users/{email}?fields={string}

Returns a user of a group to which you belong.

If you are the administrator user then all user fields are returned, otherwise only **email**, **name** and **privilege** user fields are returned.

Using cURL:

```
curl -X GET -H "authorization: Bearer YOUR-TOKEN" http://@IP/api/v1/groups/{id}/users/{email}?fields={string}
```

The optional **fields** query parameter takes a comma-separated list of user fields; only listed fields will be returned in response.

New

GET /groups/{id}/users?fields={string}

Returns the users of a group to which you belong.

If you are the administrator user then all user fields are returned, otherwise only **email**, **name** and **privilege** user fields are returned.

Using cURL:

```
curl -X GET -H "authorization: Bearer YOUR-TOKEN" http://@IP/api/v1/groups/{id}/users?fields={string}
```

The optional **fields** query parameter takes a comma-separated list of user fields; only listed fields will be returned in response.

New

PUT /groups/{id}/users/{email}

Adds a user into a group owned by you.

Using cURL:

```
curl -X PUT -H "authorization: Bearer YOUR-TOKEN" http://@IP/api/v1/groups/{id}/users/{email}
```

New

PUT /groups/{id}/users

Adds users into a group owned by you.

The optional **users** body parameter specifies users to add as a comma-separated list of emails; note that by not providing this parameter it means all available users are added into the group.

Using cURL:

```
curl -X PUT -H "authorization: Bearer YOUR-TOKEN" --data '{"users": "EMAILS"}' http://@IP/api/v1/groups/{id}/users
```

New

DELETE /groups/{id}/users/{email}

Removes a user from a group owned by you.

Using cURL:

```
curl -X DELETE -H "authorization: Bearer YOUR-TOKEN" http://@IP/api/v1/groups/{id}/users/{email}
```

New

DELETE /groups/{id}/users

Removes users from a group owned by you.

The optional **users** body parameter specifies users to remove as a comma-separated list of emails; note that by not providing this parameter it means all users of the group are removed.

Using cURL:

```
curl -X DELETE -H "authorization: Bearer YOUR-TOKEN" --data '{"users": "EMAILS"}' http://@IP/api/v1/groups/{id}/users
```

7 Links between device controlling's API and ADB remote connection

Before adding the Group feature into STF, there was some confusing behaviour between device controlling's API and ADB remote connection (i.e. in particular using latest ADB versions from 1.0.40), which could raise some functional, consistency and security issues.

This problem has been reported in detail to the STF issues web page with the number **#971** (i.e. <https://github.com/openstf/stf/issues/971>) in which the inferred solution has been implemented along the proposed Group feature.

In synthesis, it is now no more possible to ADB connect to a device if this device is not previously being controlled by the user and if this device is not listening for ABD connections on its device worker port.

As a result, hereafter is the strict matching between functional actions and API calls, UI actions or automatic timeout:

- [1] taking control of the device on:
 - clicking the **Use** button of UI
 - or calling API `POST --data '{"serial":"xxx"}' /user/devices`
- [2] releasing control of the device on:
 - clicking the **Stop Using** or **Stop Automation** buttons of UI
 - or automatic timeout
 - or calling API `DELETE /user/devices/{serial}`
- [3] opening the device worker listen port and getting the Remote debug URL on:
 - clicking the **Use** button of UI
 - or calling API `POST /user/devices/{serial} remoteConnect`
- [4] closing the device worker listen port on:
 - clicking the **Stop Using** or **Stop Automation** buttons of UI
 - or automatic timeout
 - or calling API `DELETE /user/devices/{serial} remoteConnect`
 - or calling API `DELETE /user/devices/{serial}`

So, for example using API, a typical sequence of actions a user should perform to ADB connect to a device should be:

1. takes control of the device [1]: calling API `POST --data '{"serial":"xxx"}' /user/devices`
2. opens the device worker listen port and gets the Remote debug URL [3]: calling API `POST /user/devices/{serial} remoteConnect`
3. ADB connects to the device

and, a typical sequence of actions a user should perform to ADB disconnect from a device and to release it should be:

1. ADB disconnects from the device
2. releases the device and close the device worker listen port [2][4]: calling API `DELETE /user/devices/{serial}`

8 Using Swagger UI

If you want to play with STF API against your STF platform using swagger UI tool, here is a way to achieve this following below actions (i.e. on Linux OS):

- download the **swaggerapi/swagger-ui** docker image from Docker hub:

```
# docker pull swaggerapi/swagger-ui:latest
```

- put the `api_v1.yaml` file into a special folder of your STF server:

```
# cp api_v1.yaml /opt/stf/swagger
```

- create a systemd service file for swagger (i.e. `/lib/systemd/system/swagger-ui.service`):

```
[Unit]
Description=Swagger UI
After=docker.service
BindsTo=docker.service

[Service]
EnvironmentFile=/etc/environment
TimeoutStartSec=0
Restart=always
ExecStartPre=/usr/bin/docker kill %p
ExecStartPre=/usr/bin/docker rm %p
ExecStart=/usr/bin/docker run --rm \
  --name %p \
  -e "VALIDATOR_URL=null" \
  -e "SWAGGER_JSON=/foo/api_v1.yaml" \
  -p 127.0.0.1:8085:8080 \
  -v /opt/stf/swagger:/foo \
  swaggerapi/swagger-ui:latest
ExecStop=/usr/bin/docker stop -t 2 %p

[Install]
WantedBy=multi-user.target
```

- enable and start swagger-ui service:

```
# systemctl enable swagger-ui.service
# systemctl start swagger-ui.service
```

- customize your nginx configuration file with something like that:

```
...
http {
    upstream swagger_ui {
        server localhost:8085 max_fails=0;
    }
    ...
    server {
        listen 80;
        ...
        location /swaggerui/ {
            proxy_pass http://swagger_ui/;
            proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
            proxy_set_header X-Real-IP $http_x_real_ip;
        }
        ...
    }
    ...
}
```

- restart nginx service:

```
# systemctl restart nginx.service
```


From now, connect to the <http://your-stf-server-ip-address/swaggerui> URL and enjoy as illustrated in the Figure 23 below.

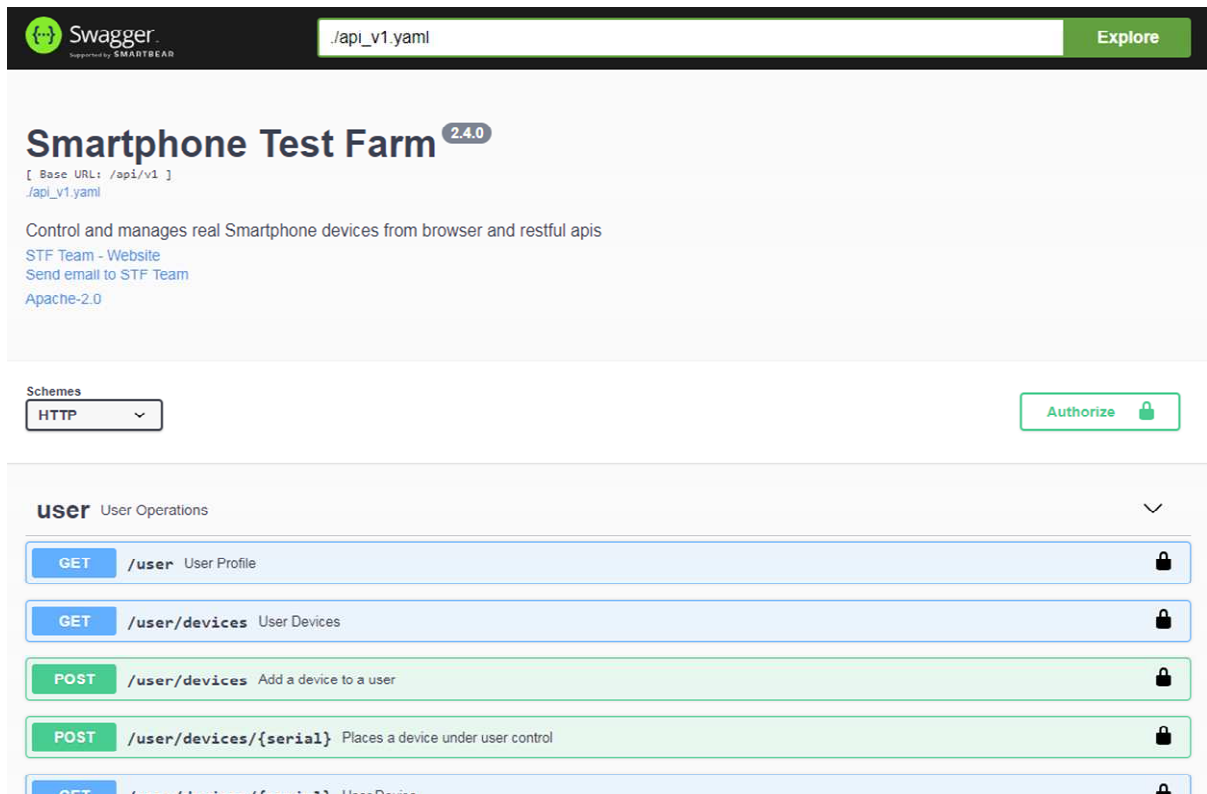


Figure 23: Swagger UI

In order to really play with the STF APIs against your platform, don't forget to authenticate by entering your access token prefixed by the "Bearer" string followed by a space, as illustrated in the Figure 24 below.

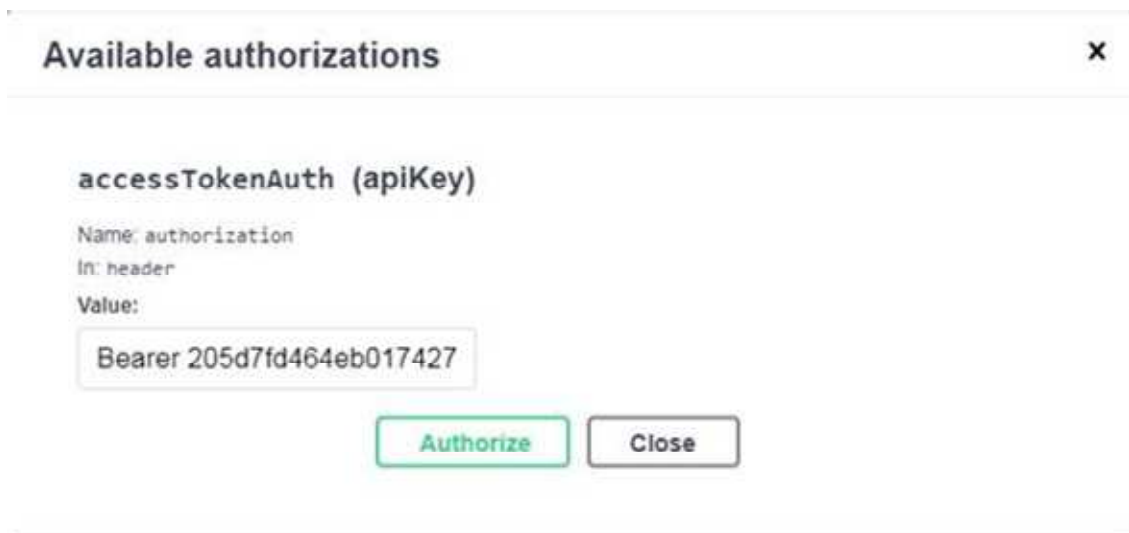


Figure 24: swagger UI authorizations

9 System configuration

Obviously, it is required to get time synchronization with accurate time servers (e.g. using NTP on Linux) on the STF machines, in order the booking & partitioning systems work fine.

10 Database migration

The Group feature introduced into STF requires some built-in objects in the database to work fine:

- the administrator user which has the following default credentials:
 - o name: **administrator**
 - o email: **administrator@fakedomain.com**

- the root standard group to which the users and devices belong, which has a default name equal to **Common**

Note you can of course override the default values of previous built-in objects by setting the following environment variables (e.g. in `/etc/environment` file on Linux) while migrating the database by launching the `stf-migrate.service` service file (cf. `DEPLOYMENT.md`):

- **STF_ROOT_GROUP_NAME** => root standard group name
- **STF_ADMIN_NAME** => administrator user name
- **STF_ADMIN_EMAIL** => administrator user email

11 Implementation details

This section is intended to STF developers only.

11.1 Architecture

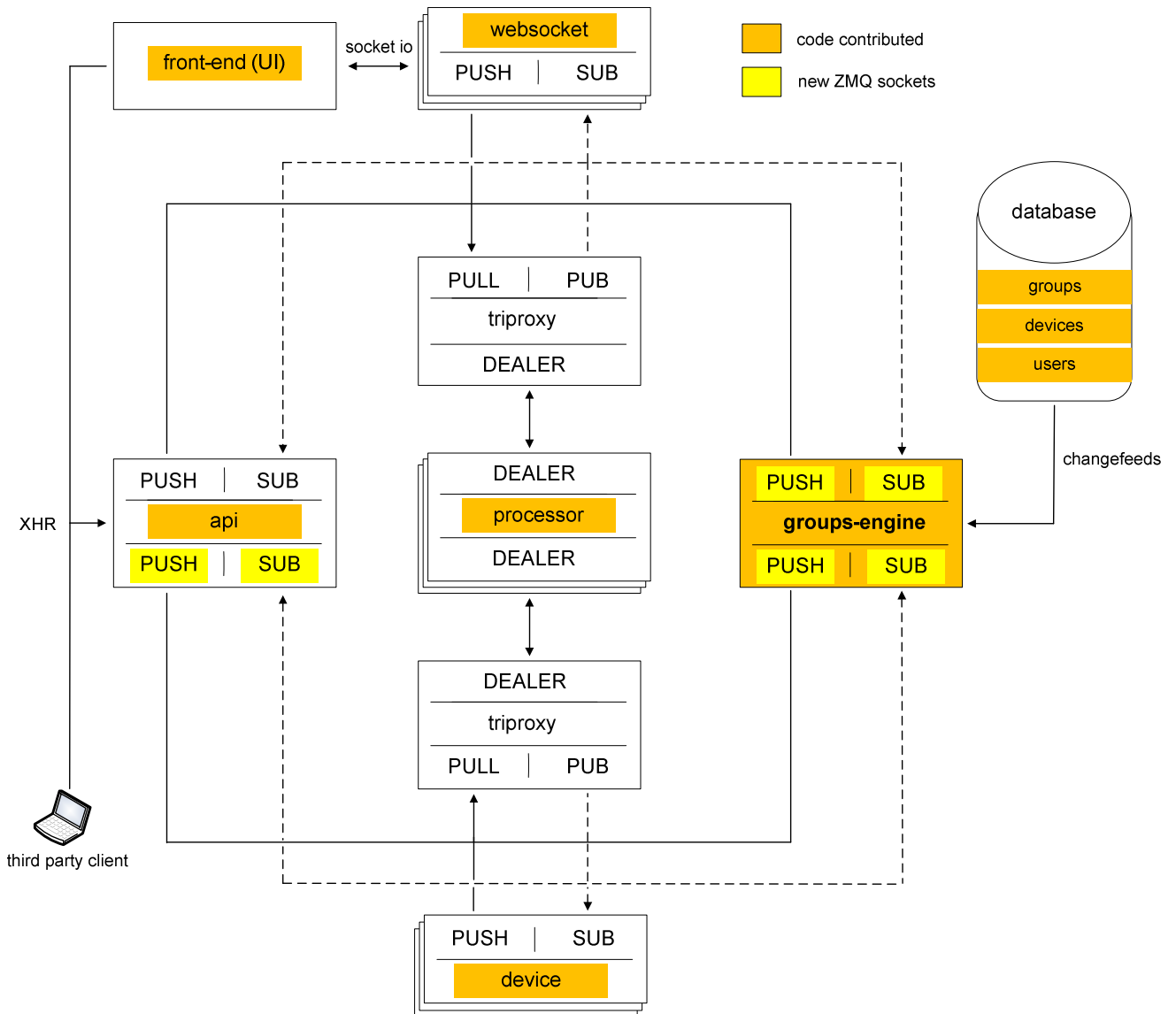


Figure 25: Architecture evolution

The STF architecture evolved mainly by adding the **groups-engine** module with direct impacts on some other modules as illustrated in the Figure 25 above.

The **groups-engine** module is made of four main functions ensuring in particular the consistency of operations ordered by the client side:

- **groups' scheduler:**
 - o triggered each second to manage lifecycle of groups: updates group state and group schedule dates, removes terminated groups, etc.
- **groups' watcher:**

- relied on *changefeeds* mechanism of **rethinkdb** database, so taking actions on group creation, updating and removing: notifies API and front-end UI, releases device control, updates device current group, etc.
- **devices' watcher:**
 - relied on *changefeeds* mechanism of **rethinkdb** database, so taking actions on device creation, updating and removing: notifies front-end UI, releases device control, etc.
- **users' watcher:**
 - relied on *changefeeds* mechanism of **rethinkdb** database, so taking actions on user creation, updating and removing: notifies front-end UI, etc.

11.2 Utilities

For development mode (e.g. robustness tests), two private STF commands have been added:

- `generate-fake-group -n=xxx => creation of xxx fake bookable groups`
 - `$ stf generate-fake-group -n=10`
- `generate-fake-user -n=xxx => creation of xxx fake users`
 - `$ stf generate-fake-user -n=10`

Note the legacy `generate-fake-device` command has been slightly modified, in particular in order generated serials are compatible while using the STF APIs (e.g. new DELETE `/devices/{serial}`).

11.3 Impacted files

Modified files: 47
Created files: 72
Total: 119

```
# git diff --shortstat master
119 files changed, 12491 insertions(+), 477 deletions(-)

# git diff --name-status master
M    LICENSE
M    lib/cli/api/index.js
A    lib/cli/generate-fake-group/index.js
A    lib/cli/generate-fake-user/index.js
A    lib/cli/groups-engine/index.js
M    lib/cli/index.js
M    lib/cli/local/index.js
M    lib/cli/migrate/index.js
M    lib/db/api.js
M    lib/db/tables.js
M    lib/units/api/controllers/devices.js
A    lib/units/api/controllers/groups.js
M    lib/units/api/controllers/user.js
A    lib/units/api/controllers/users.js
M    lib/units/api/helpers/securityHandlers.js
M    lib/units/api/index.js
M    lib/units/api/swagger/api_v1.yaml
M    lib/units/app/middleware/auth.js
M    lib/units/auth/ldap.js
M    lib/units/auth/mock.js
M    lib/units/device/plugins/connect.js
A    lib/units/groups-engine/index.js
A    lib/units/groups-engine/scheduler/index.js
A    lib/units/groups-engine/watchers/devices.js
A    lib/units/groups-engine/watchers/groups.js
A    lib/units/groups-engine/watchers/users.js
M    lib/units/processor/index.js
```

Group feature for STF by Orange SA – Version 1.0

```
M lib/units/websocket/index.js
A lib/util/apiutil.js
M lib/util/datautil.js
M lib/util/deviceutil.js
M lib/util/fakedevice.js
A lib/util/fakegroup.js
A lib/util/fakeuser.js
A lib/util/lockutil.js
A lib/util/timeutil.js
M lib/wire/wire.proto
M res/app/app.js
A res/app/components/stf/column-choice/column-choice-directive.js
A res/app/components/stf/column-choice/column-choice.css
A res/app/components/stf/column-choice/column-choice.pug
A res/app/components/stf/column-choice/index.js
M res/app/components/stf/common-ui/index.js
A res/app/components/stf/common-ui/modals/generic-modal/generic-modal-service.js
A res/app/components/stf/common-ui/modals/generic-modal/generic-modal-spec.js
A res/app/components/stf/common-ui/modals/generic-modal/generic-modal.pug
A res/app/components/stf/common-ui/modals/generic-modal/index.js
M res/app/components/stf/common-ui/modals/index.js
A res/app/components/stf/common-ui/pagination/index.js
A res/app/components/stf/common-ui/pagination/pagination-directive.js
A res/app/components/stf/common-ui/pagination/pagination-filter.js
A res/app/components/stf/common-ui/pagination/pagination-service.js
A res/app/components/stf/common-ui/pagination/pagination.css
A res/app/components/stf/common-ui/pagination/pagination.pug
M res/app/components/stf/device/device-service.js
M res/app/components/stf/device/enhance-device/enhance-device-service.js
A res/app/components/stf/devices/devices-service.js
A res/app/components/stf/devices/index.js
A res/app/components/stf/groups/groups-service.js
A res/app/components/stf/groups/index.js
M res/app/components/stf/tokens/access-token-service.js
M res/app/components/stf/user/user-service.js
A res/app/components/stf/users/index.js
A res/app/components/stf/users/users-service.js
A res/app/components/stf/util/common/common-service.js
A res/app/components/stf/util/common/index.js
M res/app/control-panes/control-panes-controller.js
M res/app/device-list/column/device-column-service.js
M res/app/device-list/column/index.js
M res/app/device-list/device-list-controller.js
M res/app/device-list/stats/device-list-stats-directive.js
A res/app/group-list/group-list-controller.js
A res/app/group-list/group-list.css
A res/app/group-list/group-list.pug
A res/app/group-list/groups/groups.pug
A res/app/group-list/index.js
A res/app/group-list/stats/group-quota-stats.pug
A res/app/group-list/stats/group-stats.pug
M res/app/menu/index.js
M res/app/menu/menu-controller.js
M res/app/menu/menu.pug
A res/app/settings/devices/devices-controller.js
A res/app/settings/devices/devices-spec.js
A res/app/settings/devices/devices.css
A res/app/settings/devices/devices.pug
A res/app/settings/devices/index.js
A res/app/settings/general/date-format/date-format-controller.js
A res/app/settings/general/date-format/date-format.pug
A res/app/settings/general/date-format/index.js
A res/app/settings/general/email-address-separator/email-address-separator-controller.js
A res/app/settings/general/email-address-separator/email-address-separator.pug
A res/app/settings/general/email-address-separator/index.js
M res/app/settings/general/general.pug
M res/app/settings/general/index.js
A res/app/settings/groups/conflicts/conflicts.pug
A res/app/settings/groups/devices/devices.pug
A res/app/settings/groups/filters/available-objects-filter.js
A res/app/settings/groups/filters/group-objects-filter.js
A res/app/settings/groups/groups/groups-controller.js
A res/app/settings/groups/groups-spec.js
A res/app/settings/groups/groups.css
A res/app/settings/groups/groups.pug
A res/app/settings/groups/index.js
A res/app/settings/groups/schedule/schedule.pug
A res/app/settings/groups/users/users.pug
```

Group feature for STF by Orange SA – Version 1.0

```
M    res/app/settings/index.js
M    res/app/settings/settings-controller.js
A    res/app/settings/settings.css
A    res/app/settings/users/index.js
A    res/app/settings/users/users-controller.js
A    res/app/settings/users/users-spec.js
A    res/app/settings/users/users.css
A    res/app/settings/users/users.pug
M    res/auth/ldap/scripts/signin/index.js
M    res/auth/ldap/scripts/signin/signin-controller.js
M    res/auth/ldap/scripts/signin/signin.pug
M    res/auth/mock/scripts/signin/index.js
M    res/auth/mock/scripts/signin/signin-controller.js
M    res/auth/mock/scripts/signin/signin.pug
```