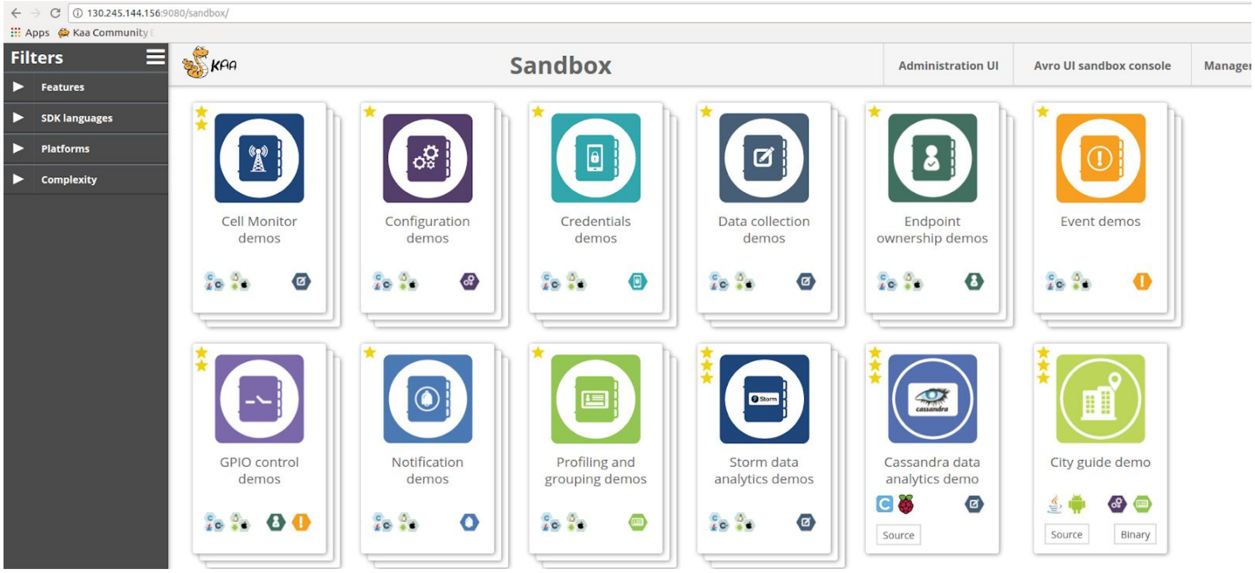
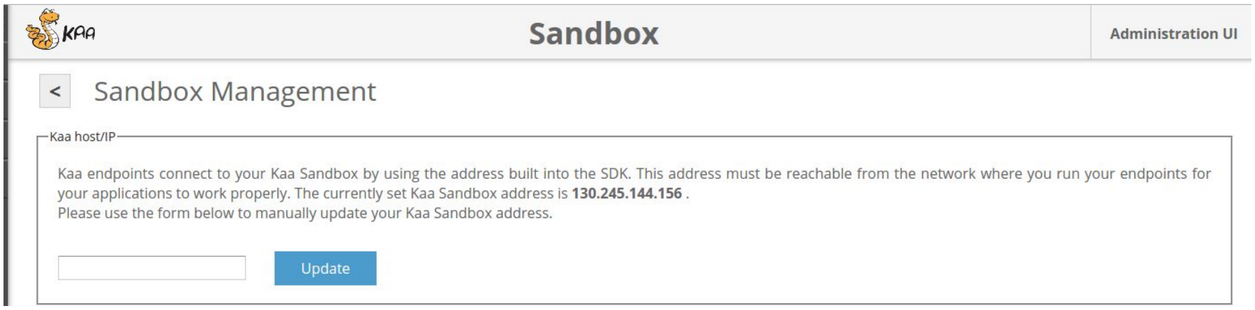
SpecSense Installation and Deployment Guide

To deploy SpecSense, we need the Kaa IoT framework. The Kaa Sandbox, image is available at https://www.kaaproject.org/community-edition/

-Download the sandbox from the above link. You might have to create an account to be able to download the sandbox image.
-Import it on VirtualBox.
-Once imported, change the network settings for that VM and use a Bridged Adapter for it, since we want to be able to access the Kaa server from other machines in the network.
-Now start the VM. This will start the Kaa Server.
-Once the Kaa server is started, use http://<ip_address>:9080/sandbox/ to go to the Kaa portal. (It might say site not reachable, but allow a few minutes after the VM is up for the web page to load)



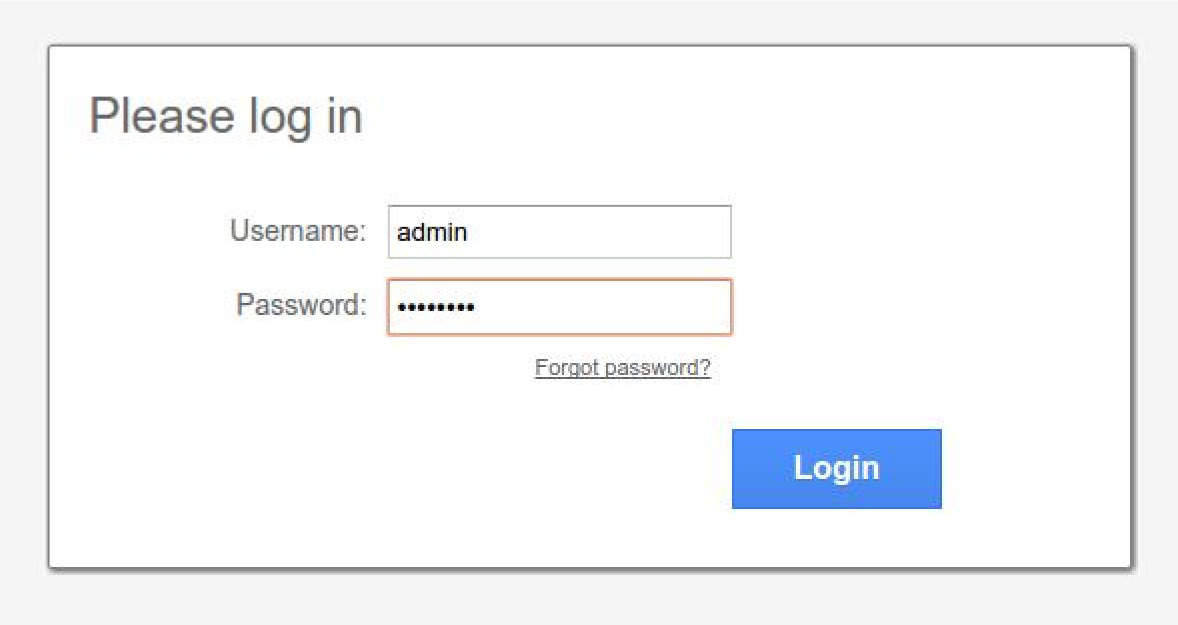
-Once there, click on the Management tab, to check if Kaa has the correct IP address set. This IP address is important since it will be embedded in SDK and will be used to send data from the endpoints to the server.



Kaa endpoints connect to your Kaa Sandbox by using the address built into the SDK. This address must be reachable from the network where you run your endpoints for your applications to work properly. The currently set Kaa Sandbox address is **130.245.144.156** .
Please use the form below to manually update your Kaa Sandbox address.

-Once the IP address is correct, go back to the above page and click on Administration UI. This will take you to a login page.

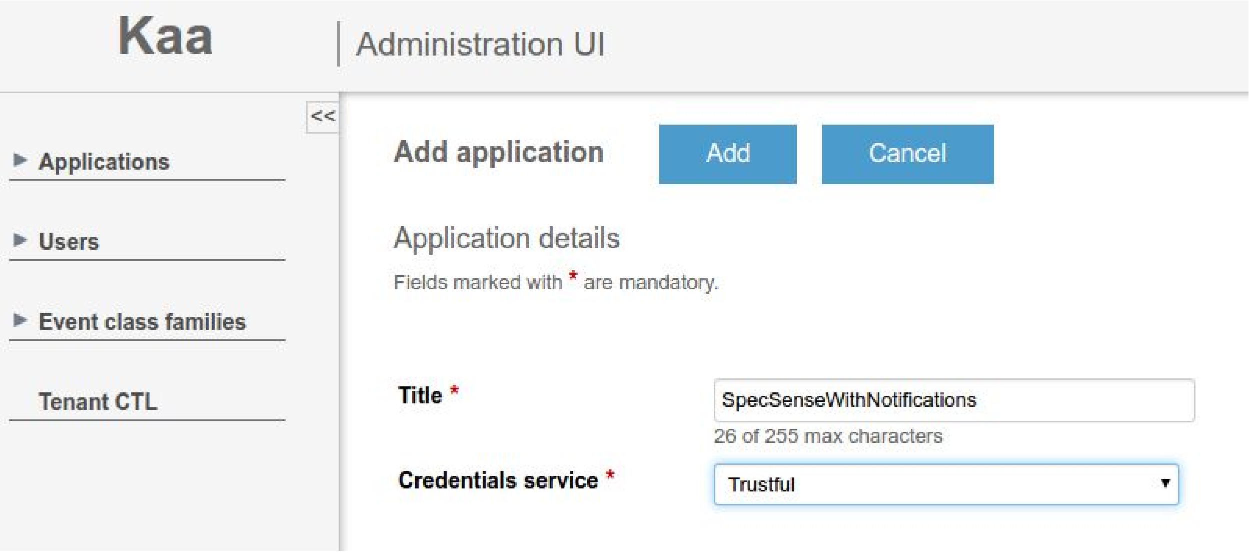
-If we have not yet created our SpecSense application on Kaa, login as **admin/admin123** and click on Create New Application and give the application a name and select Trustful for Credentials service. And finally click on Add Application. This will create the SpecSense application.



Please log in

Username: admin

Password: •••••••••



Add application

Application details

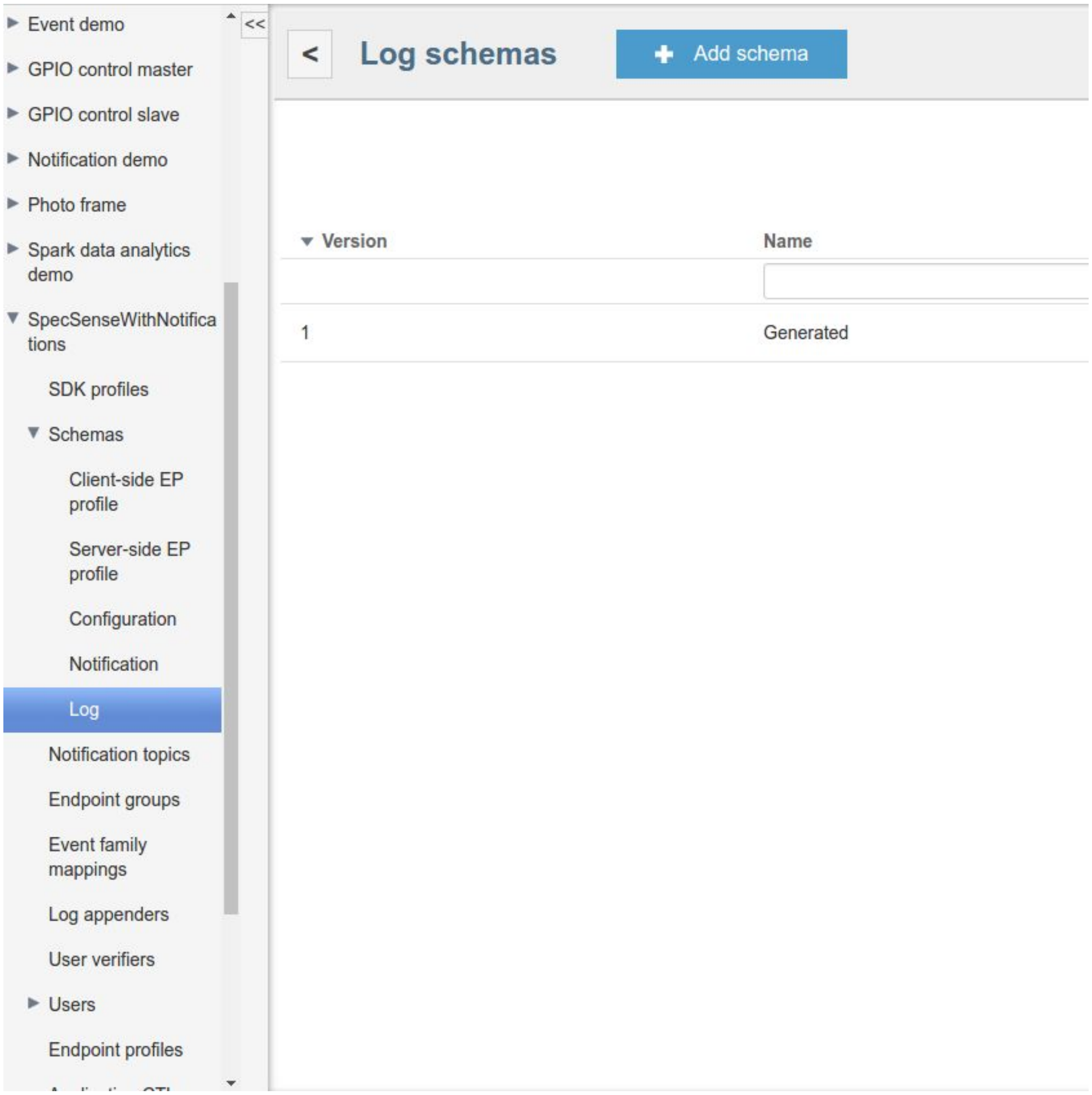
Fields marked with * are mandatory.

Title * SpecSenseWithNotifications

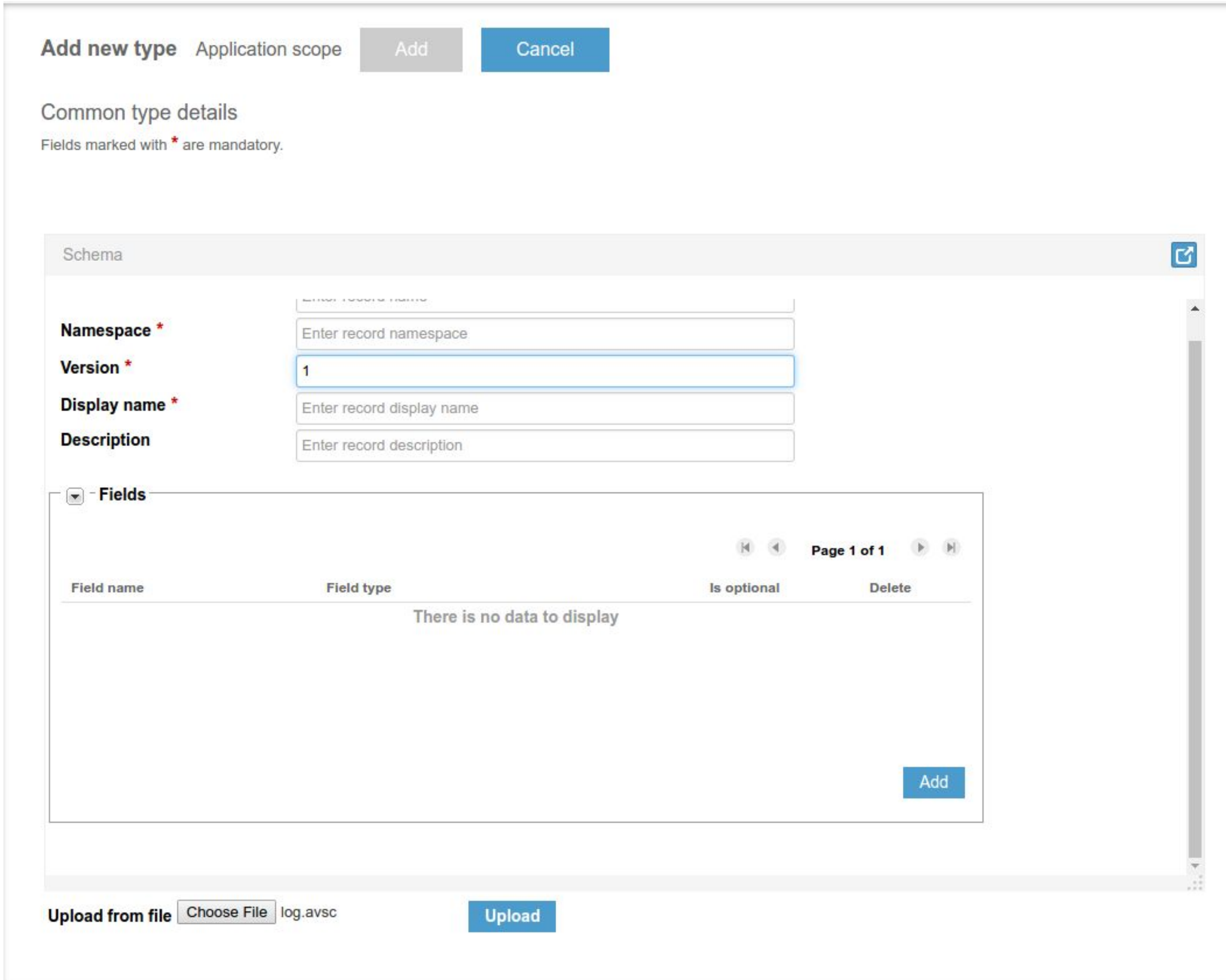
26 of 255 max characters

Credentials service * Trustful

-Now logout from admin user and login again with **devuser/devuser123**. This is where we can customize our application schemas and other application related configurations, and also generate the SDKs.
-Once logged in, go to the SpecSense application which we created.
-Navigate to Schemas->Log. Here, we will create our log schema by clicking on **Add Schema** and then clicking on **Create New Type**. Instead of filling in all the fields manually, you can use the **log.avsc** file available on the Github repository (https://github.com/Wings-Lab/SpectrumSensingWithNotifications) for this project in **avro** directory. Scroll down on this screen and upload the **log.avsc** file and say Upload. This will upload the file and extract all the required fields for our logs namely, power, frequency, iq, and node number. Once all the fields are extracted, click on **Add**. Now the new Schema is added.



▼ Version	Name
1	Generated



Common type details

Fields marked with * are mandatory.

Enter record name

Namespace * Enter record namespace

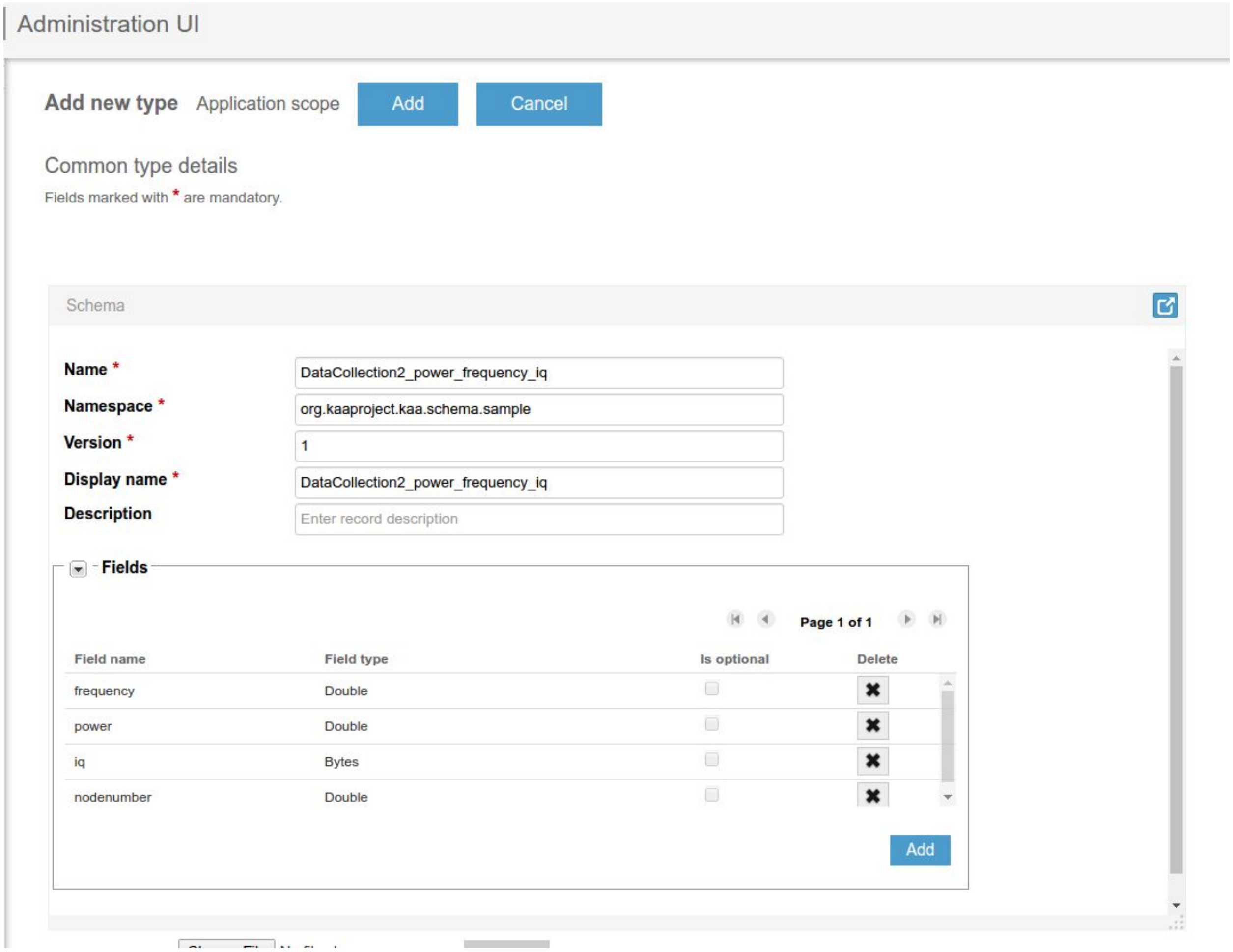
Version * 1

Display name * Enter record display name

Description Enter record description

Field name	Field type	Is optional	Delete
There is no data to display			

Upload from file Choose File log.avsc



Common type details

Fields marked with * are mandatory.

Name * DataCollection2_power_frequency_iq

Namespace * org.kaaproject.kaa.schema.sample

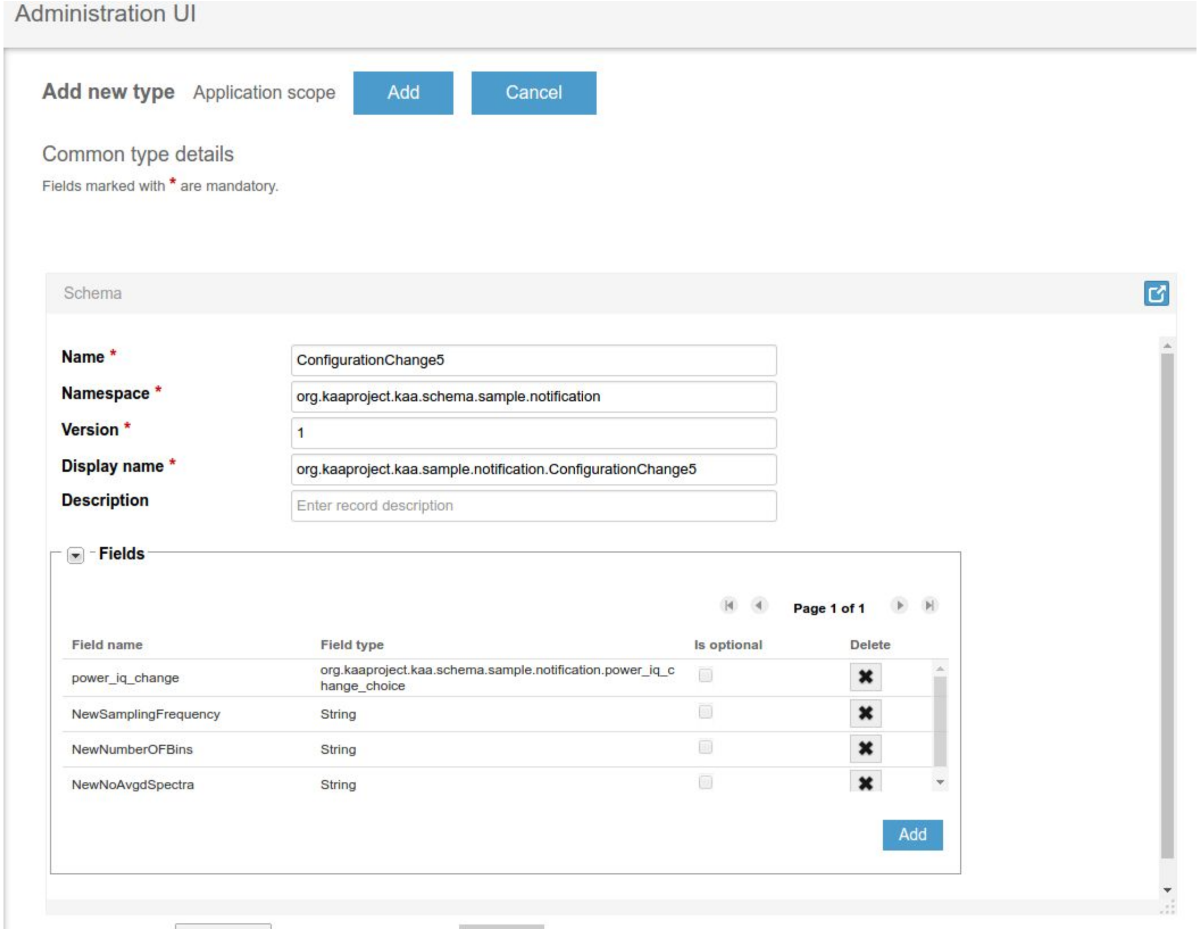
Version * 1

Display name * DataCollection2_power_frequency_iq

Description Enter record description

Field name	Field type	Is optional	Delete
frequency	Double	□	✖
power	Double	□	✖
iq	Bytes	□	✖
nodenumber	Double	□	✖

-Now navigate to Schemas->Notification. Similar to log schema, we will import the notification schema with the **notification.avsc** file available in **avro** directory in the Github repository (https://github.com/Wings-Lab/SpectrumSensingWithNotifications). Click on Add Schema. The notification schema will be added.



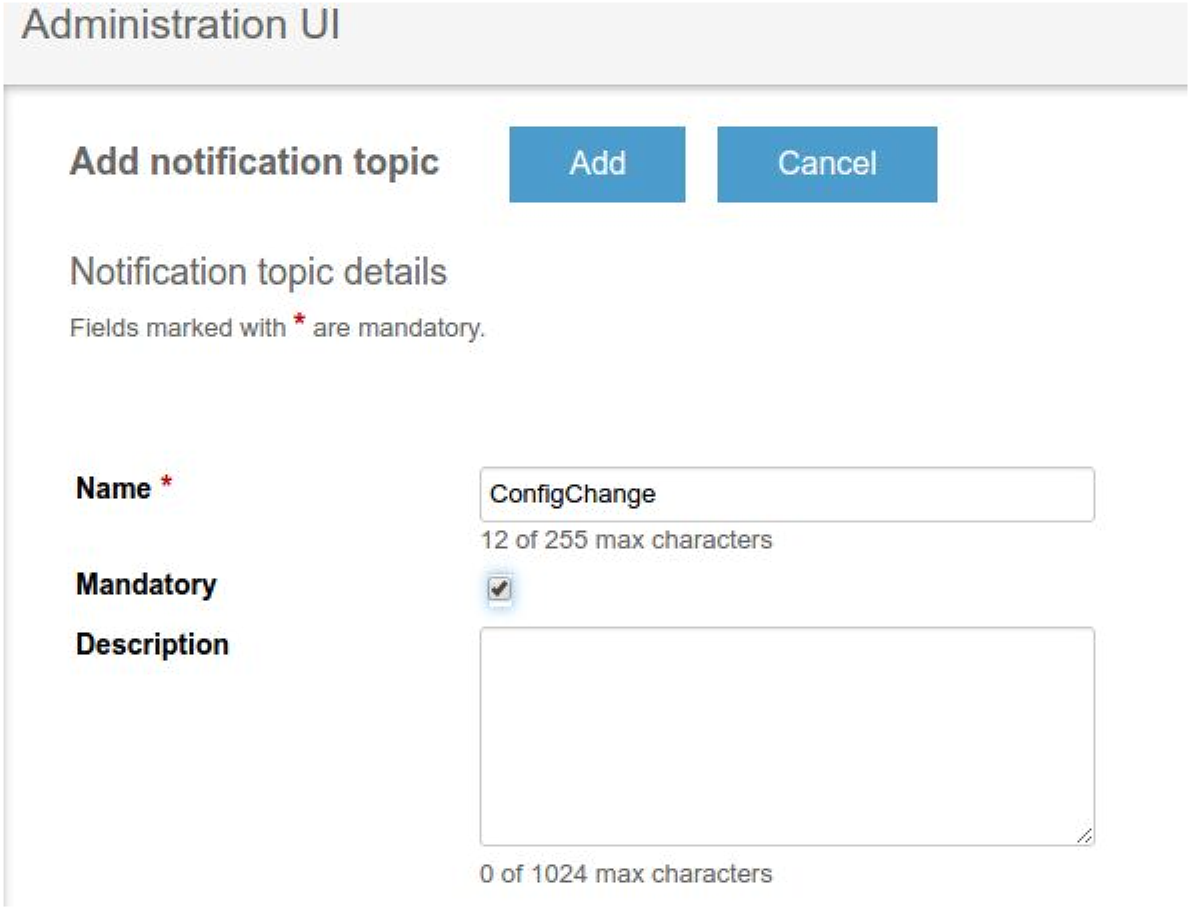
Common type details

Fields marked with * are mandatory.

Name *	ConfigurationChange5
Namespace *	org.kaaproject.kaa.schema.sample.notification
Version *	1
Display name *	org.kaaproject.kaa.sample.notification.ConfigurationChange5
Description	Enter record description

Field name	Field type	Is optional	Delete
power_iq_change	org.kaaproject.kaa.schema.sample.notification.power_iq_change_choice	□	✖
NewSamplingFrequency	String	□	✖
NewNumberOFBins	String	□	✖
NewNoAvgdSpectra	String	□	✖

-Once the schemas are added, we need to add a notification topic. Navigate to **Notification Topics** and add a topic and make it mandatory, so that all endpoints are subscribed to the notifications for this topic by default.



Administration UI

Add notification topic

Notification topic details

Fields marked with * are mandatory.

Name * ConfigChange

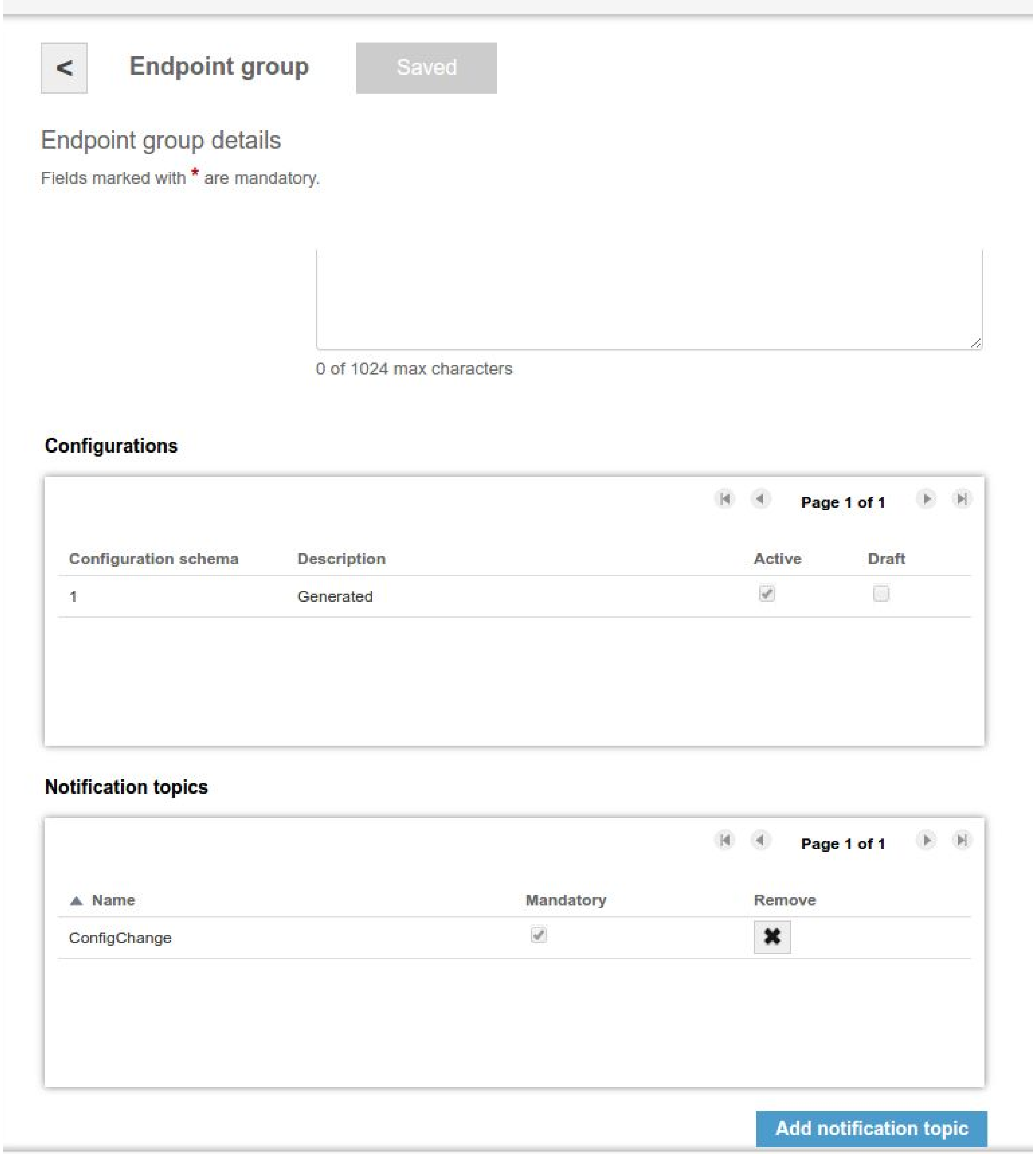
12 of 255 max characters

Mandatory ☑

Description

0 of 1024 max characters

-After this, navigate to Endpoint Groups and click on All. Here, scroll down to Notification Topics and add the topic created above to the ‘All’ Endpoint Group.



Endpoint group details

Fields marked with * are mandatory.

0 of 1024 max characters

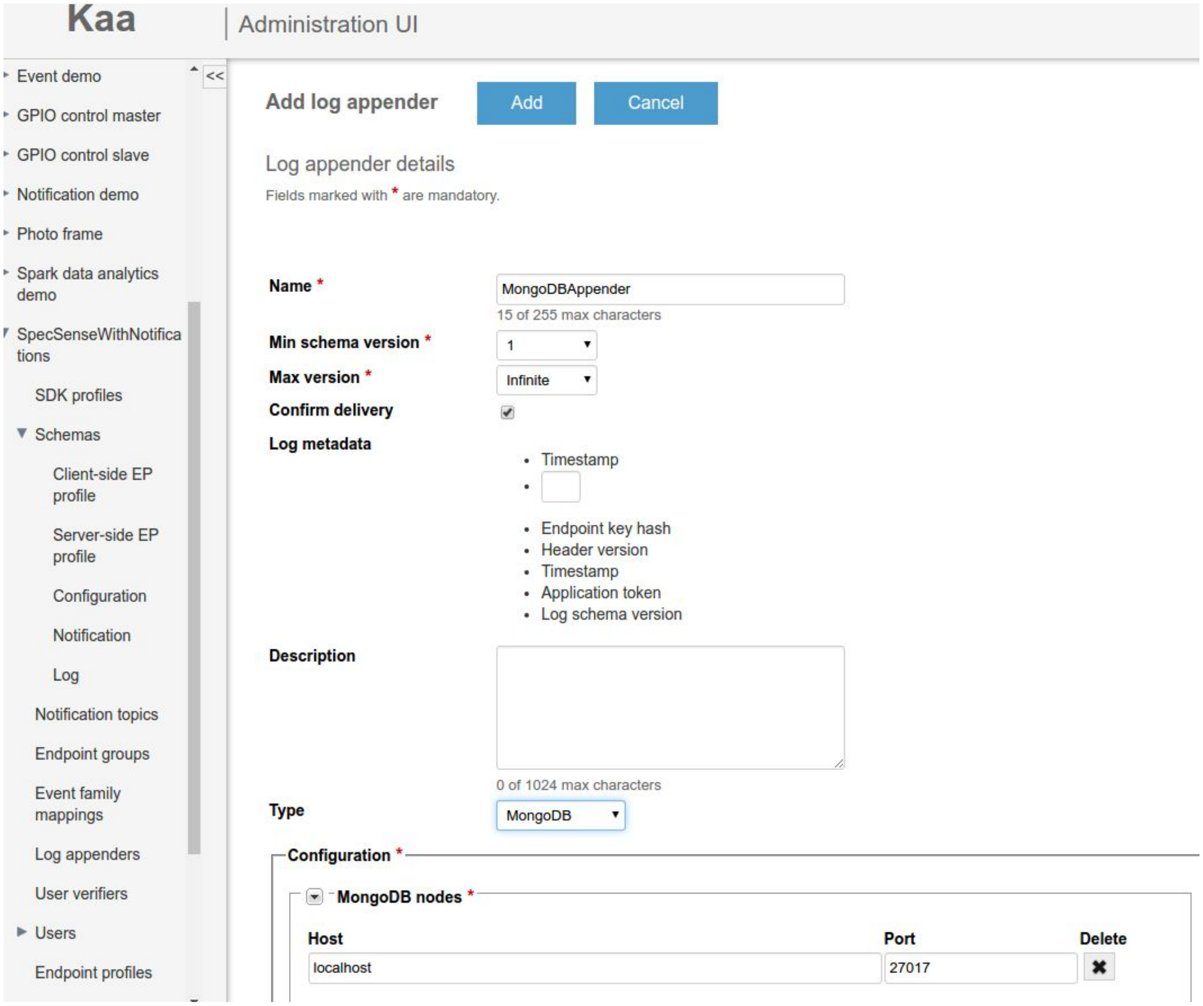
Configurations

Configuration schema	Description	Active	Draft
1	Generated	☑	□

Notification topics

▲ Name	Mandatory	Remove
ConfigChange	☑	✖

-After this, navigate to Log Appenders. This is where we add our log appender i.e. the database where our logs will be stored. Here, name the new log appender, select the additional attribute ‘timestamp’ and select type as **‘MongoDB’** (this means that our logs will be stored in a MongoDB database on the Kaa server)



Add log appender

Log appender details

Fields marked with * are mandatory.

Name * MongoDBAppender
15 of 255 max characters

Min schema version * 1

Max version * Infinite

Confirm delivery ☑

Log metadata

- Timestamp
-

- Endpoint key hash
- Header version
- Timestamp
- Application token
- Log schema version

Description

0 of 1024 max characters

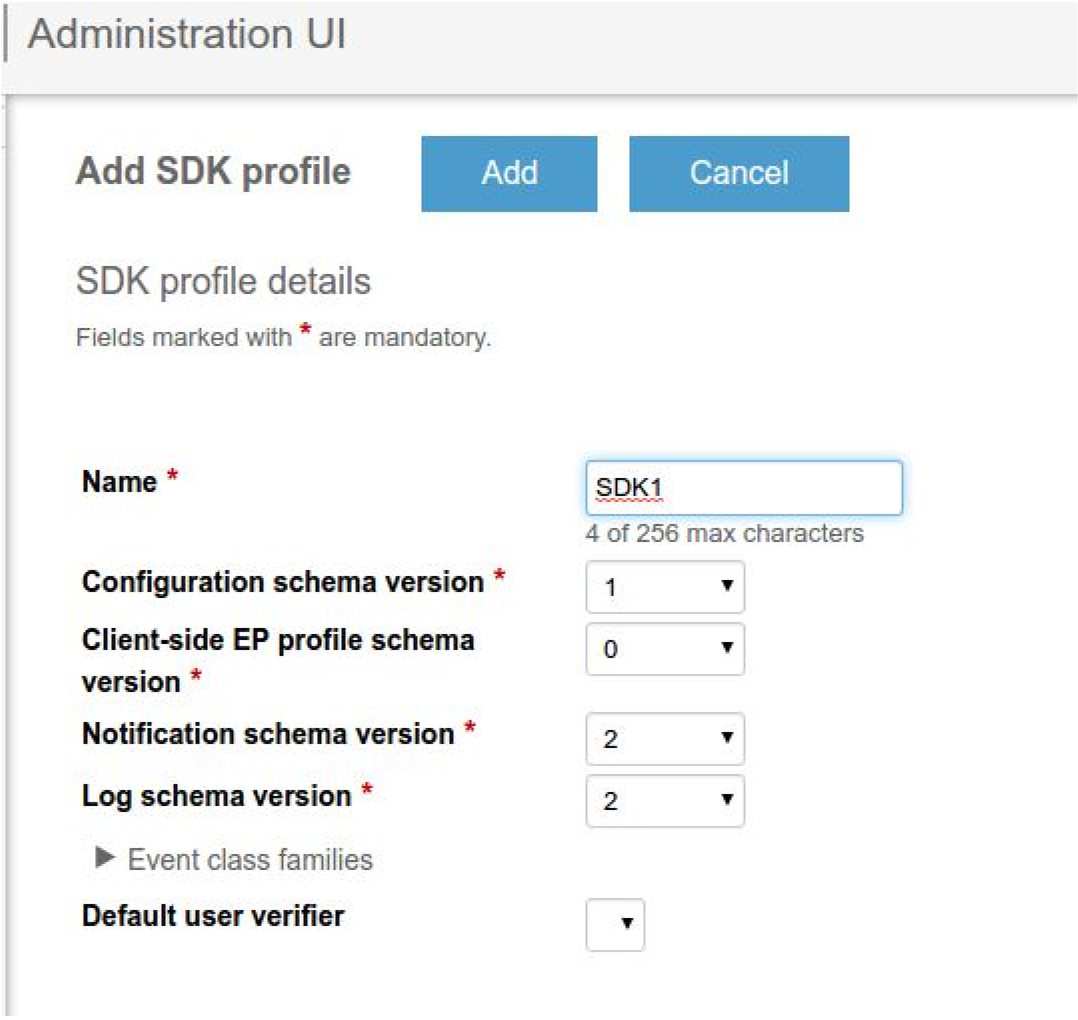
Type MongoDB

Configuration *

MongoDB nodes *

Host	Port	Delete
localhost	27017	✖

-Once all the above steps are done, we need to generate the SDK for the application. To do this, navigate to **SDK Profiles** and click on **Create New**. The latest versions for the notification and log schemas will already be selected. Name the SDK and click on **Add**.

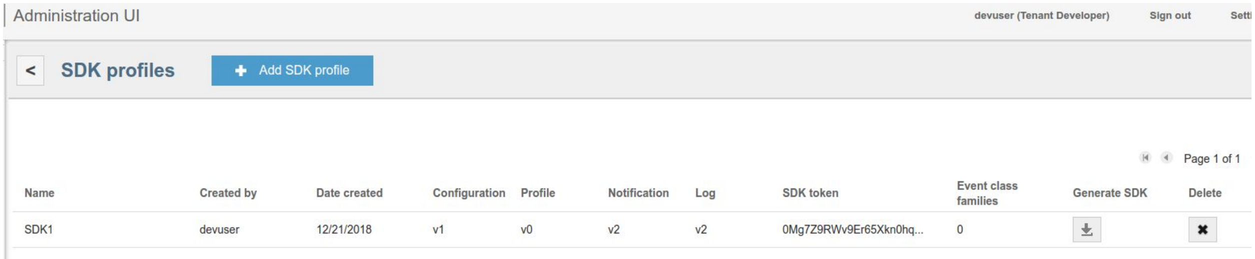


SDK profile details

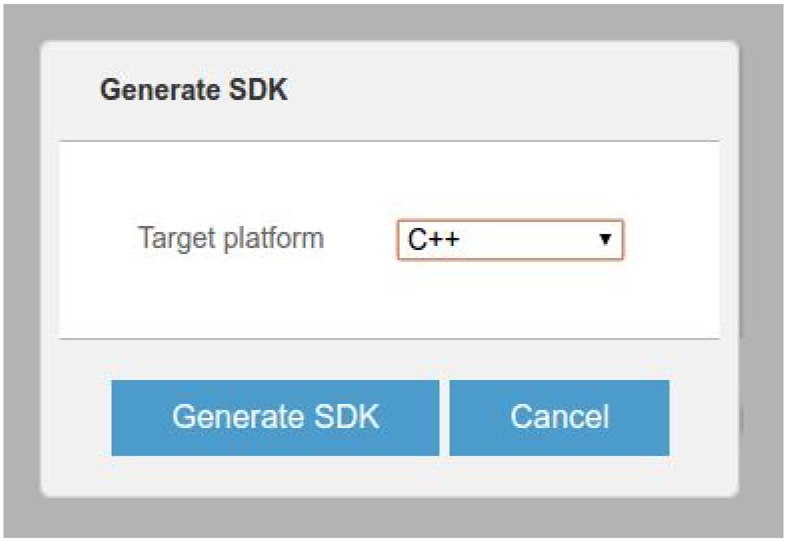
Fields marked with * are mandatory.

Name *	SDK1 4 of 256 max characters
Configuration schema version *	1
Client-side EP profile schema version *	0
Notification schema version *	2
Log schema version *	2
▶ Event class families	
Default user verifier	

-Now we can download the newly created SDK. Click on the download sign on the SDK. Choose the language as C++. Now the SDK will be downloaded as a zip.



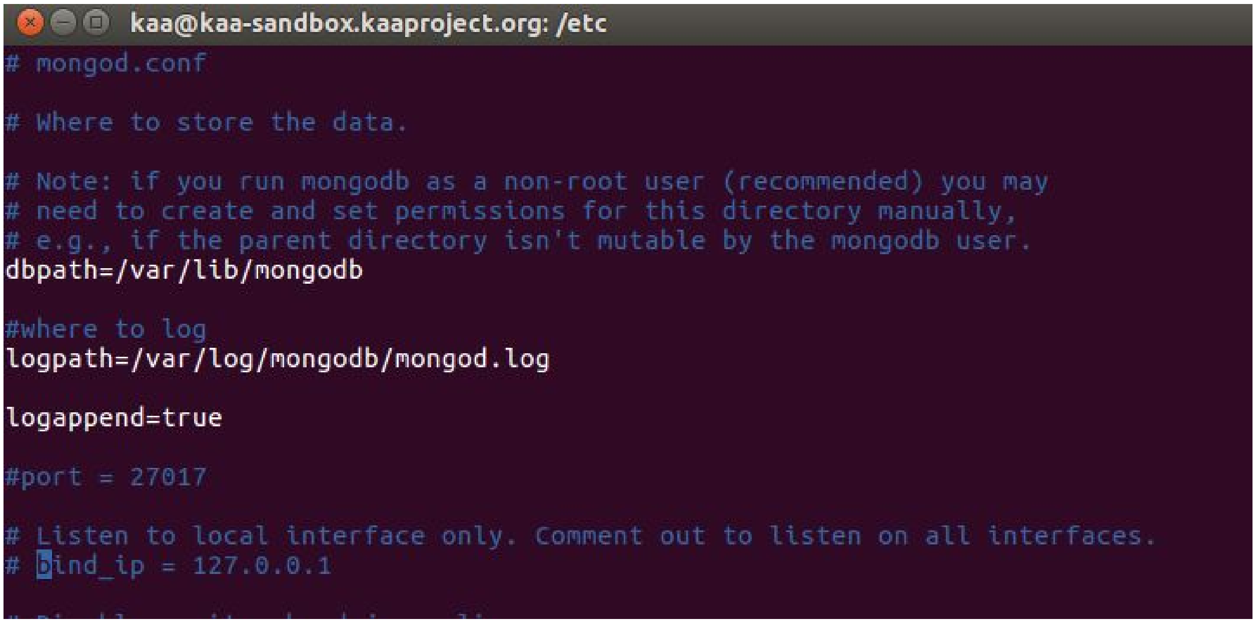
Name	Created by	Date created	Configuration	Profile	Notification	Log	SDK token	Event class families	Generate SDK	Delete
SDK1	devuser	12/21/2018	v1	v0	v2	v2	0Mg7Z9RWv9Er65Xkn0hq...	0	⬇	✖


Generate SDK

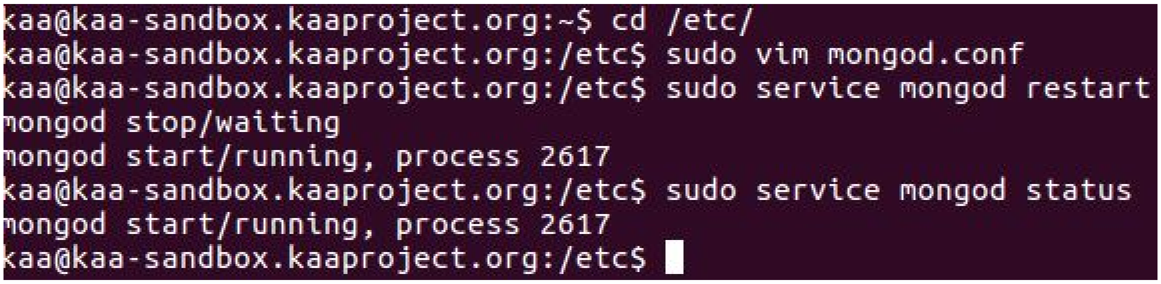
Target platform C++ ▾

-Extract that zip and copy all the contents of the zip to the endpoints (Odroid boards) using scp or anything similar.
-The SDK contents should be copied to the **kaa** directory in the main application directory (The contents of the main application directory are available on the Github repository (https://github.com/Wings-Lab/SpectrumSensingWithNotifications), and they have to be copied to the Odroid boards before we put the SDKs on them) on the Odroid boards. Make sure that you replace the existing contents of the **kaa** directory with the new ones.
-Once the SDK is copied to the kaa directory, ssh into the Odroid boards one by one and navigate to the build folder. Here, delete all the contents using **rm -rf *. ** Then run the command **cmake -DKAA_MAX_LOG_LEVEL=3 ..**
-This will generate the files required for the build with the new SDK files. Now run the command **make -j4.** This will build the entire client application for us, and an executable named ‘kaa-app’ will be created.
-After the build is successful, use the command **cp ../rtl_power_fftw ./** and copy the **rtl_power_fftw** executable to the build folder.

Note: Before running the code on the Odroid boards, we need to enable MongoDB available on the Kaa server to accept connections from all IP addresses. To do this, ssh into the kaa sandbox server using **ssh kaa@<ip_address>** and navigate to **/etc/**. Here, open the file mongod.conf using **sudo vim mongod.conf** and comment the line which says **bind_ip = 127.0.0.1**. By commenting out this line, we are telling the MongoDB server on Kaa to accept connections from all interfaces and not just the local interface. After making this change in /etc/mongod.conf, we need to to restart the MongoDB service on the server with the command **sudo service mongod restart**. Confirm that MongoDB is running with the command **sudo service mongod status**.

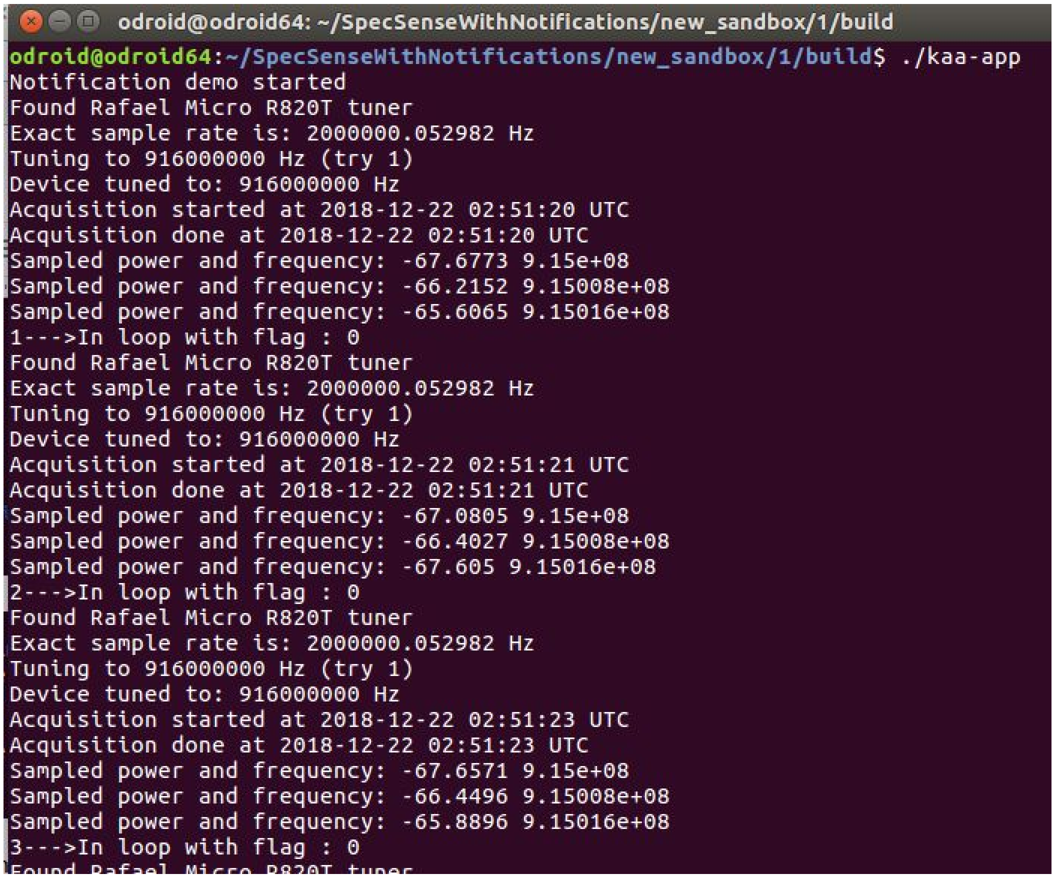


```
# mongod.conf

# Where to store the data.

# Note: if you run mongodb as a non-root user (recommended) you may
# need to create and set permissions for this directory manually,
# e.g., if the parent directory isn't mutable by the mongodb user.
dbpath=/var/lib/mongodb

#where to log
logpath=/var/log/mongodb/mongod.log

logappend=true

#port = 27017

# Listen to local interface only. Comment out to listen on all interfaces.
# bind_ip = 127.0.0.1
```



```
kaa@kaa-sandbox.kaaproject.org:~$ cd /etc/
kaa@kaa-sandbox.kaaproject.org:/etc$ sudo vim mongod.conf
kaa@kaa-sandbox.kaaproject.org:/etc$ sudo service mongod restart
mongod stop/waiting
mongod start/running, process 2617
kaa@kaa-sandbox.kaaproject.org:/etc$ sudo service mongod status
mongod start/running, process 2617
kaa@kaa-sandbox.kaaproject.org:/etc$
```

-Now our application is ready to run. Just run **./kaa-app** and the application will start sensing data and sending it to the server. The console will print something like this:


odroid@odroid64: ~/SpecSenseWithNotifications/new_sandbox/1/build

```
odroid@odroid64:~/SpecSenseWithNotifications/new_sandbox/1/build$ ./kaa-app
Notification demo started
Found Rafael Micro R820T tuner
Exact sample rate is: 2000000.052982 Hz
Tuning to 916000000 Hz (try 1)
Device tuned to: 916000000 Hz
Acquisition started at 2018-12-22 02:51:20 UTC
Acquisition done at 2018-12-22 02:51:20 UTC
Sampled power and frequency: -67.6773 9.15e+08
Sampled power and frequency: -66.2152 9.15008e+08
Sampled power and frequency: -65.6065 9.15016e+08
1--->In loop with flag : 0
Found Rafael Micro R820T tuner
Exact sample rate is: 2000000.052982 Hz
Tuning to 916000000 Hz (try 1)
Device tuned to: 916000000 Hz
Acquisition started at 2018-12-22 02:51:21 UTC
Acquisition done at 2018-12-22 02:51:21 UTC
Sampled power and frequency: -67.0805 9.15e+08
Sampled power and frequency: -66.4027 9.15008e+08
Sampled power and frequency: -67.605 9.15016e+08
2--->In loop with flag : 0
Found Rafael Micro R820T tuner
Exact sample rate is: 2000000.052982 Hz
Tuning to 916000000 Hz (try 1)
Device tuned to: 916000000 Hz
Acquisition started at 2018-12-22 02:51:23 UTC
Acquisition done at 2018-12-22 02:51:23 UTC
Sampled power and frequency: -67.6571 9.15e+08
Sampled power and frequency: -66.4496 9.15008e+08
Sampled power and frequency: -65.8896 9.15016e+08
3--->In loop with flag : 0
Found Rafael Micro R820T tuner
```

-You can then run the python scripts **data_rate_computation.py, drop_db.py, and periodic_drop.py** to check the number of records received at the database, to delete the database and to periodically run the delete script respectively. **For this, make sure that the collection name in the above files is the correct one from the portal**. Use python2 to run the above files.