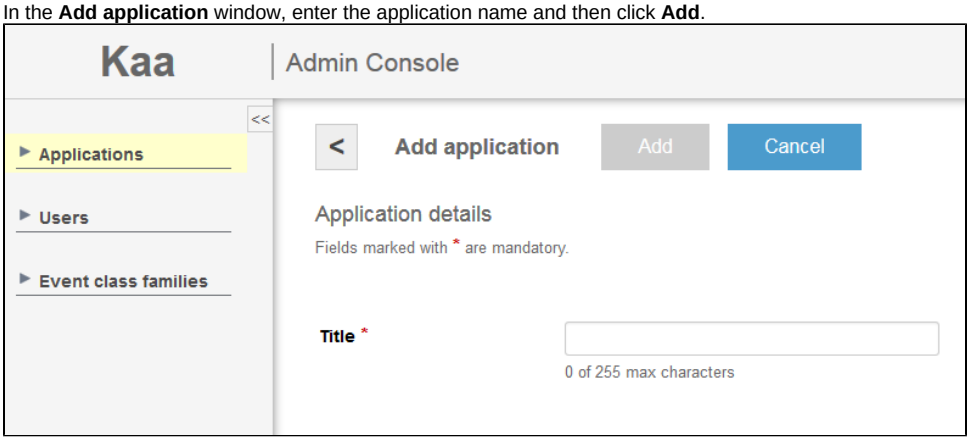
Your first Kaa application

To register a new application within a fresh Kaa server installation, you need to create users with the *tenant administrator* and *tenant developer* roles. T he tenant administrator is responsible for creating new applications in Kaa, and the tenant developer configures and generates SDKs for those applications. We suggest that you use Kaa Sandbox, which has a tenant administrator and tenant developer users already created.

Adding application

To add an application, proceed as follows:

1. Open the Kaa admin UI in your browser (http://127.0.0.1:8080) and log in as a tenant administrator (user/password: admin/admin123).
2. Select **Applications** on the navigation panel on the left side and, in the **Applications** window that opens, click **Add application.**
3. In the **Add application** window, enter the application name and then click **Add**.



After the application has been added, you may log out. We will not be using the tenant administrator role in this guide anymore.

Creating notification schema

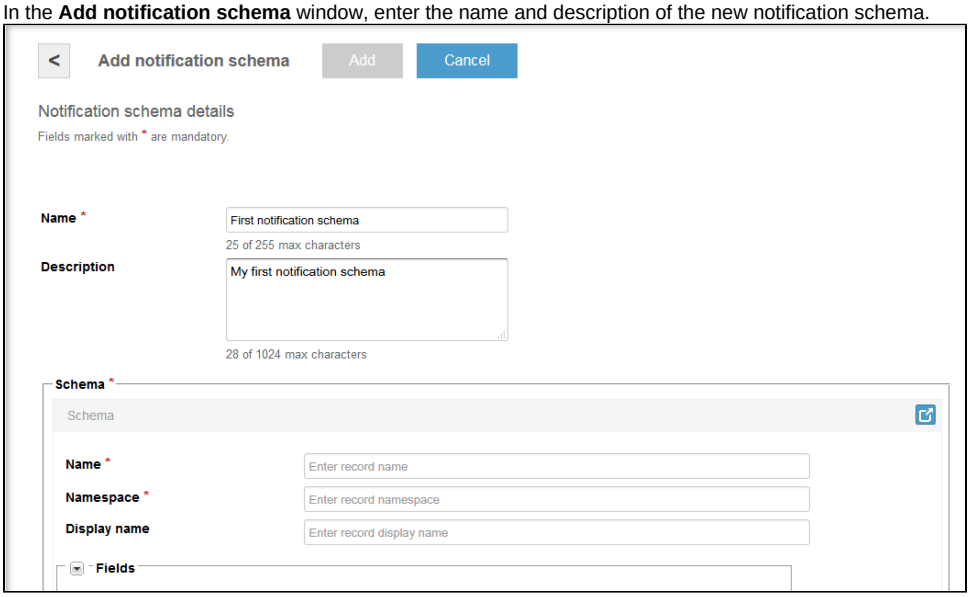
The application that you have created in the previous step already includes the default versions of the profile, configuration, notification and log schemas ready for use. However, in this sample application, we will use a custom notification schema for demonstration purposes. To create and upload the schema, proceed as follows:

1. Create the *schema.json* file on your PC with the following schema definition:

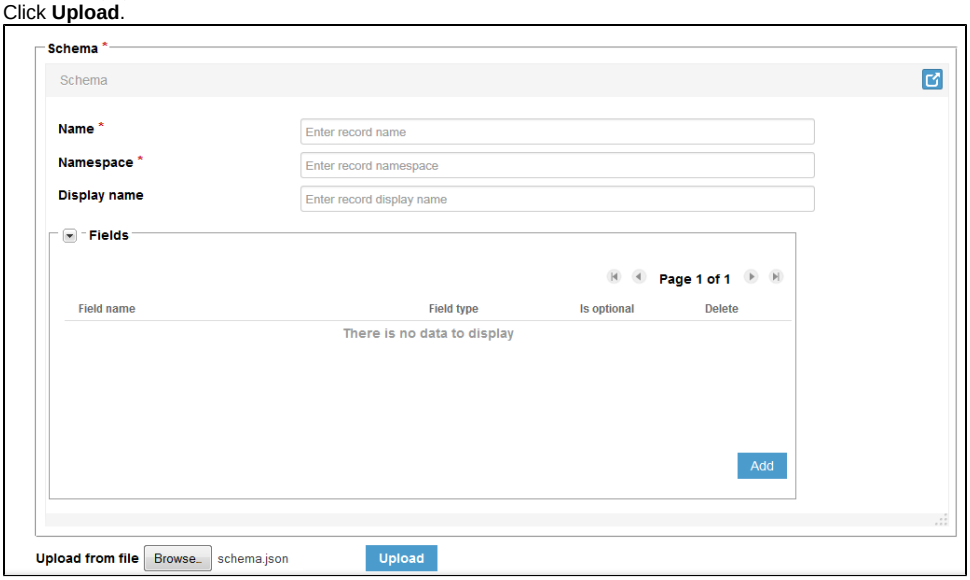
```
{
    "type": "record",
    "name": "Notification",
    "namespace": "org.kaaproject.kaa.schema.example",
    "fields": [
        {
            "name": "message",
            "type": "string"
        }
    ]
}
```

2. Open the admin UI in your browser (http://127.0.0.1:8080) and log in as a tenant developer (user/password: devuser/devuser123).
3. Open the relevant **Notification schemas** window (**Applications => My First Kaa Application => Schemas => Notification**) and click **Add schema**.

4. In the **Add notification schema** window, enter the name and description of the new notification schema.



5. Scroll down and use the **Upload from file** function to find the previously created json file with the schema. Alternatively, you can use the Schema Avro UI form to create the schema.
6. Click **Upload**.



7. Click **Add** at the top of the window.

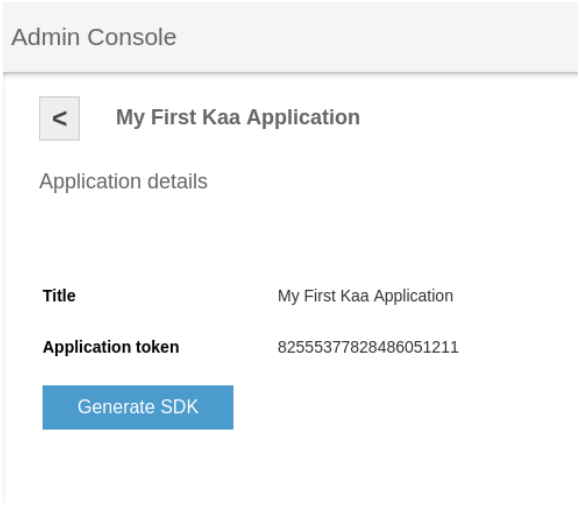
As a result of this operation you will see two notification schemas in the list:

In this screenshot, version 2.0 is the notification schema that was just created. We will use this version for the SDK generation in the next step.

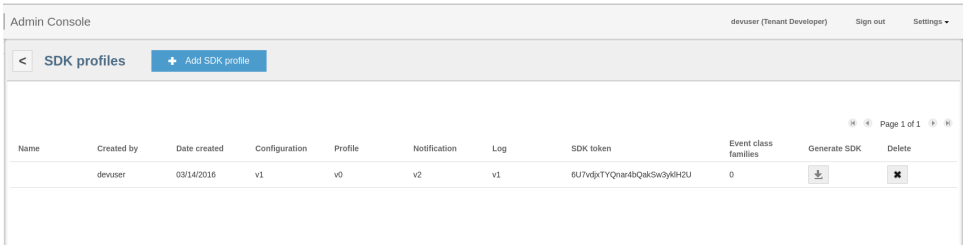
Generating SDK

To generate an SDK for the new application, proceed as follows:

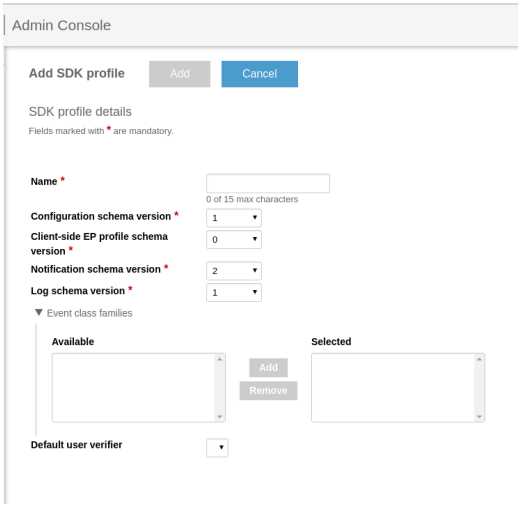
1. Select the **My First Kaa Application** application and click **Generate SDK.**



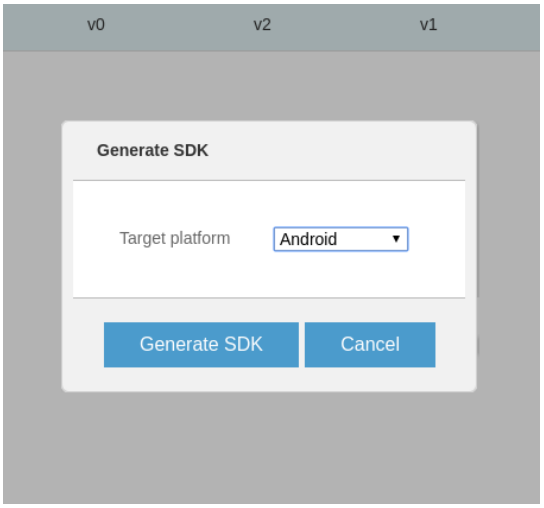
2. Click **Add SDK profile**.



3. In the **Add SDK profile** window, fill in the fields as shown in the following screenshot and then click **Add**.



4. Click **Generate SDK** for corresponding SDK profile. In the **Generate SDK** window select the target platform for your SDK and click **Generate SDK**.



After the SDK is generated, you will be presented with a window asking you to save a .jar file with the generated SDK (for Java) or an archive with the generated SDK (for C, C++ or Objective-C). Specify the file name and location on your computer and then click **Save**. The SDK is now downloaded to your computer.

Note that in this example we are generating the SDK based on the default configuration, profile, and log schemas. These schemas are automatically populated during the creation of the application. If necessary, you can overwrite them using Admin UI.

Sample client application

Once you have downloaded the SDK, you can use it in your sample project. The following code block illustrates a simple desktop application that will receive notifications from the Kaa server and display them on the console.

> **NOTE:** After generating the C/C++/Objective-C SDKs, you need to build them before creating the application.

You can find the project source code (including Java, C, C++ and Objective-C) in the attached archive.

- **For Java**, the project is built using Apache Maven. Note that the downloaded SDK must be placed into the *lib* folder in order for the build to work. Import this project into your IDE as a Maven project and launch the *NotificationSystemTestApp* application. Once the application is launched, you will see the following output:

```
[...  INFO  NotificationSystemTestApp]  Kaa SDK client started!
```

- **For C, C++ and Objective-C**, place the downloaded SDK into the *libs* folder and then run *build.sh* or *build.bat* script depending on your platform. Launch the *demo_client* application. Once the application is launched, you will see the following output:

```
Kaa SDK client started!
```

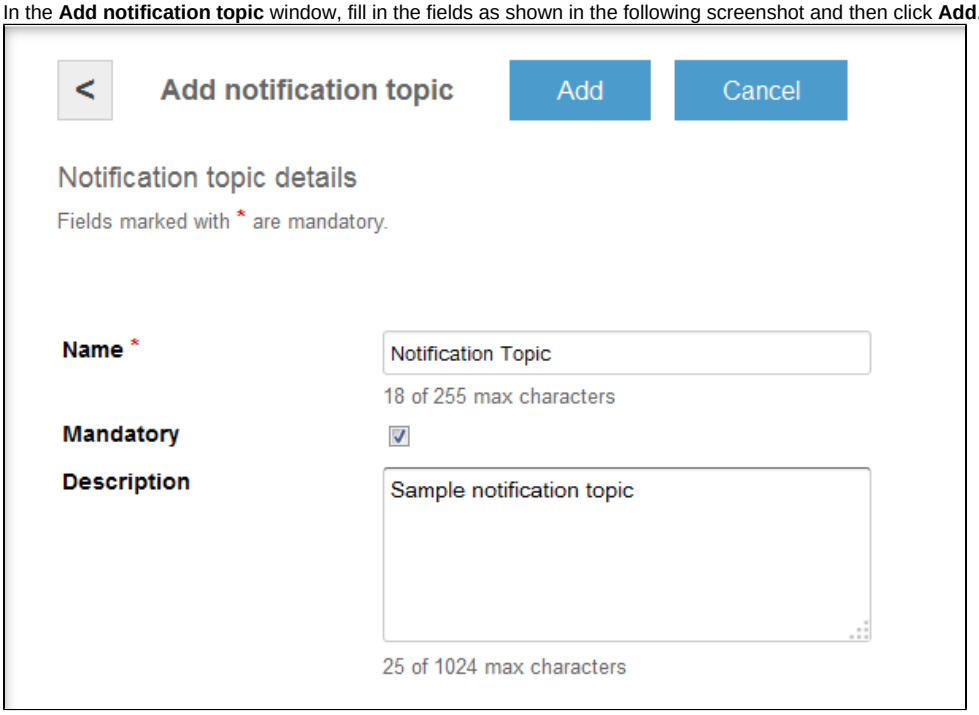
Creating notification topic

To send your first Kaa notification, you need to create a notification topic and assign this topic to the default endpoint group.

To create a notification topic, proceed as follows:

1. Open the relevant **Notification topics** window (**Applications => My First Kaa Application => Notification topics**) and click **Add notification topic**.

2. In the **Add notification topic** window, fill in the fields as shown in the following screenshot and then click **Add**.

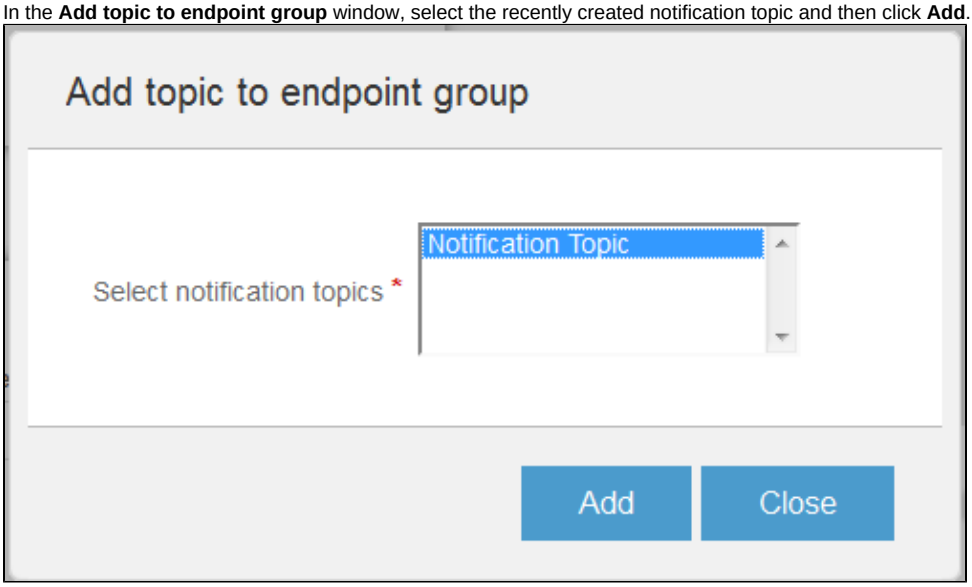


NOTE: We set the topic as mandatory in order to automatically subscribe the client application to notifications on this topic.

Once the topic is created, we will assign it to the default endpoint group, which contains all endpoints, including endpoints with our application.

To assign a notification topic to the default endpoint group, proceed as follows:

1. In the relevant **Endpoint groups** window (**Applications=>My First Kaa Application=>Endpoint groups**), select the *All* group.
2. In the **Endpoint group details** window, click **Add notification topic** at the bottom of the window.
3. In the **Add topic to endpoint group** window, select the recently created notification topic and then click **Add**.



After the topic is added to the endpoint group, you will see the following output in the application:

```
[...  INFO  NotificationSystemTestApp]  Topic list updated!
[...  INFO  NotificationSystemTestApp]  Received topic with id X and name Notification Topic
```

This is the first update from the Kaa server that provides the client with the information about the changes in the topic list. Now you can send notifications on this topic, and they will be delivered to your application.

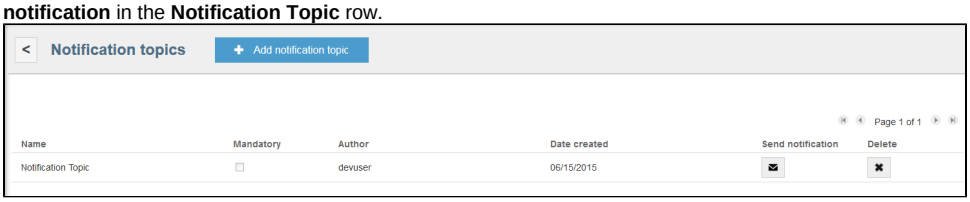
Creating notification

To create a notification, proceed as follows:

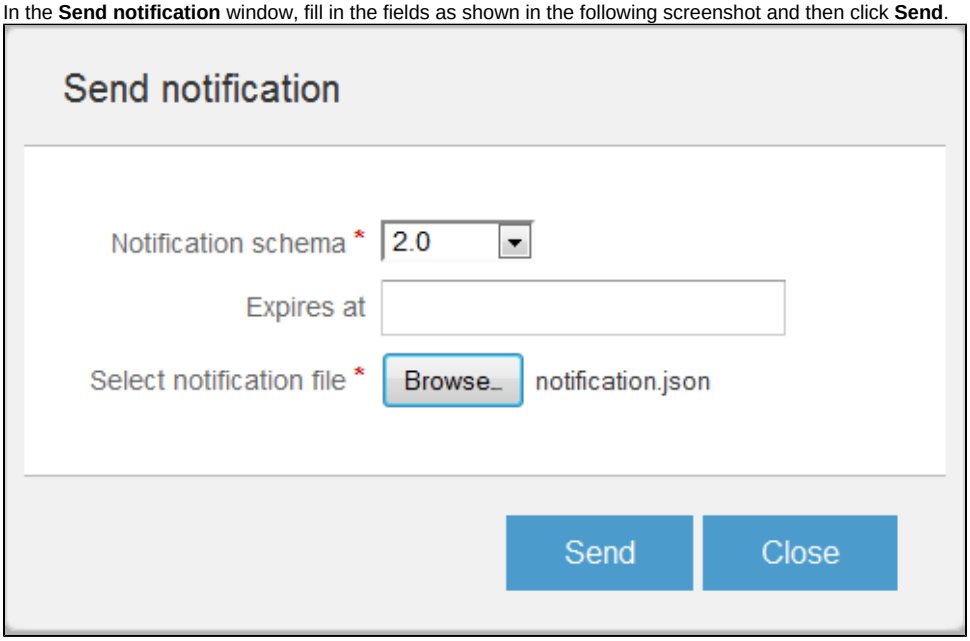
1. Create a *notification.json* file on your PC with the following contents:

```
{"message":"Hello from Kaa!"}
```

2. Open the relevant **Notification topics** window (**Applications => My First Kaa Application => Notification topics**) and click **Send notification** in the **Notification Topic** row.



3. In the **Send notification** window, fill in the fields as shown in the following screenshot and then click **Send**.



Once the notification is sent, you can see the following output in the application:

```
[...  INFO  NotificationSystemTestApp]  Received notification {"message": "Hello from Kaa!"} for topic with
id X
```

Congratulations with your first Kaa application!

Next steps

To create a real-world IoT solution, you will most likely need to implement more features into your application. Kaa provides you with practically everything you might need. The following overview will help you grasp the scope of Kaa capabilities as well as get familiar with the essential documentation, such as Programming guide and Administration UI guide.

Profiling and grouping

During a new endpoint registration, Kaa creates an associated *endpoint profile* for the endpoint. An endpoint profile is basically some meaningful information about the endpoint which may be useful for specific applications. Profiles may contain things like an OS version, amount of RAM, average battery life, type of network connection, device operation mode – virtually anything. An endpoint profile structure in Kaa is configured using a *client-side endpoint profile schema*. Based on the defined profile schema, Kaa generates an object model to operate against the client side and handles data marshaling all the way to the database. Whenever the client updates its profile information, the endpoint SDK automatically sends these updates to the server as soon as the connection becomes available.

For programming practice, see collecting endpoint profiles.

The information collected in an endpoint's profile can be used to group endpoints into independently managed entities called *endpoint groups*. On the back end, Kaa provides a profile filtering language for defining the criteria for group membership. An endpoint can belong to any number of groups. Grouping endpoints can be used, for example, to send targeted notifications or adjust software behavior by applying group-specific configuration overrides.

For programming practice, see using endpoint groups.

Events

Kaa allows for delivery of *events*, which are structured messages, across endpoints. When endpoints register with the Kaa server, they communicate which event types they are able to generate and receive. Kaa allows endpoints to send events either to virtual “chat rooms” or to individual endpoints. Events can even be delivered across applications registered with Kaa – making it possible to quickly integrate and enable interoperability between endpoints running different applications. Some examples are: a mobile application that controls house lighting, a car’s GPS that communicates with the home security system, a set of integrated audio systems from different vendors that deliver a smooth playback experience as you walk from one room to another. Kaa events are implemented in a generic, abstract way, using non-proprietary schema definitions that ensure identical message structures. The schema provides independence from any specific functionality implementation details.

For programming practice, see messaging across endpoints.

Collecting data

Kaa provides rich capabilities for collecting and storing structured data from endpoints. A typical use-case is collecting various types of logs: performance, user behavior, exceptional conditions, etc.

Using a set of pre-packaged server-side *log appenders*, the Kaa server is able to store records to a filesystem, a variety of big data platforms (Hadoop, MongoDB, Cassandra, Oracle NoSQL etc.), or submit them directly to a streaming analytics system. It is also possible to create a custom log appender.

The structure of the collected data is flexible and defined by the *log schema*. Based on the log schema defined for the Kaa application, Kaa generates an object model for the records and the corresponding API calls in the client SDK. Kaa also takes care of data marshalling, managing temporary data storage on the endpoint, and uploading data to the Kaa server.

For programming practice, see collecting data from endpoints.

Using notifications

Kaa uses *notifications* to distribute structured messages, posted within *notification topics*, from the server to endpoints. A notification structure is defined by a corresponding notification schema.

Endpoint are subscribed to notification topics, which can be either mandatory or optional. Access to notification topics is automatically granted according to the endpoint’s group membership.Notifications can be sent either to every endpoint subscribed to a topic or to an individual endpoint.

Notifications can be assigned expiration timestamps to prevent their delivery after a certain period of time.

For programming practice, see using notifications.

Distributing operational data

Kaa allows you to perform operational data updates, such as configuration data updates, from the Kaa server to endpoints. This feature can be used for centralized configuration management, content distribution, etc. Since Kaa works with structured data and constraint types, it guarantees data integrity.

The Kaa server monitors the database for changes and distributes updates to endpoints in the incremental form, thus ensuring efficient bandwidth use. The endpoint SDK performs data merging and persistence, as well as notifies the client code about the specific changes made to the data. As a result, the client application knows exactly where in the data structure the changes occurred and can be programmed to react accordingly.

Based on the endpoint’s group membership, it is possible to control what data is available to the endpoint. This is achieved by applying group-specific data overrides, which make it possible to adjust the behavior of the client application based on operational conditions or usage patterns, fine-tune the algorithms according to feedback, implement gradual feature roll-out, A/B testing, etc.

For programming practice, see distributing data to endpoints.

Further reading

Use the following guides and references to make the most of Kaa.

Guide	What it is for
Design reference	Use this reference to learn about features and capabilities of Kaa (Endpoint profiling, Events, Notifications, Logging, and other features).
Kaa Sandbox	Use this guide to try out Kaa in a private environment with demo applications.
Programming guide	Use this guide to create advanced applications with Kaa
Installation guide	Use this guide to install and configure Kaa either on a single Linux node or in a cluster environment.
Contribute to Kaa	Use this guide to learn how to contribute to Kaa project and which code/documentation style conventions we adhere to.