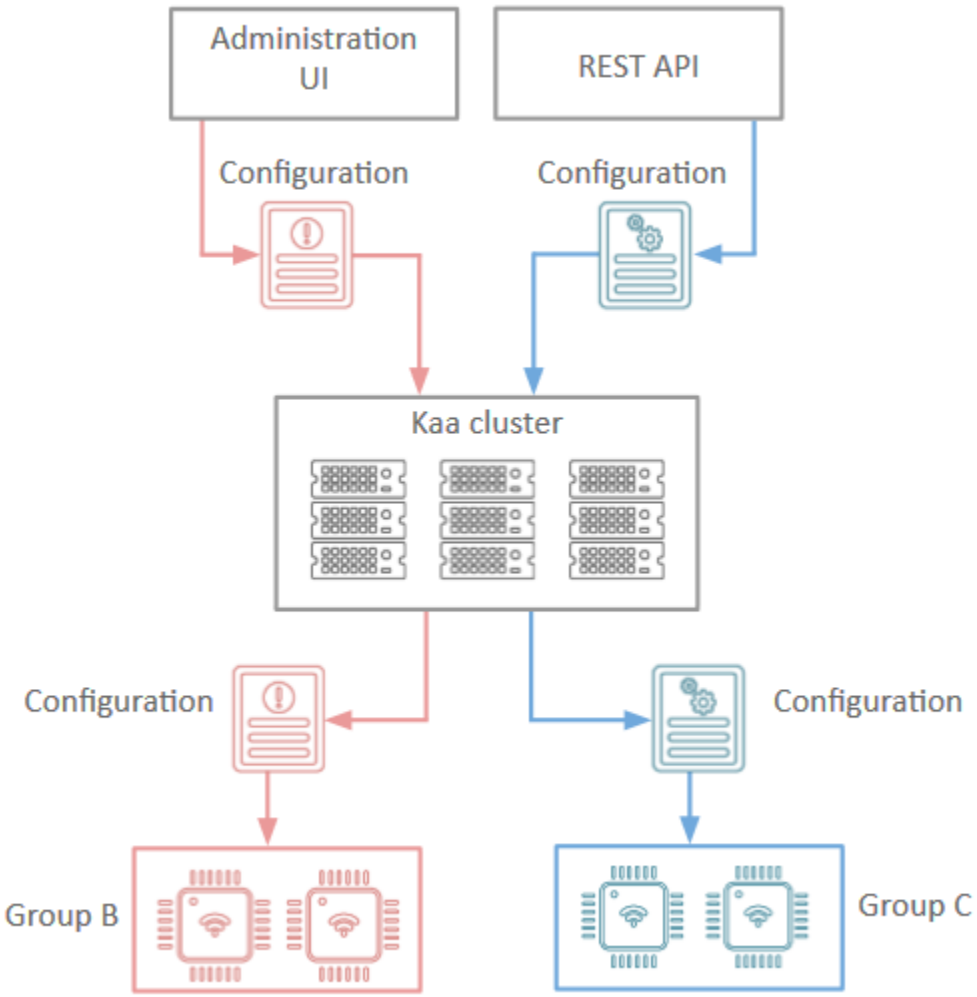
Distributing data to endpoints

The Kaa configuration subsystem supplies endpoints with the structured configuration data of custom complexity that is managed through the Kaa server. The configuration data itself is structured according to the configuration schema specified by the user beforehand. Both a configuration schema and configuration data can be modified on the server using Admin UI or REST API and then uploaded to relevant applications.

Basic architecture

The following diagram illustrates basic entities and data flows in scope of the configuration management:

- Configuration is generated based on the configuration schema created by the developer for the application
- The user or admin distributes new configuration using either Admin UI or REST API
- It is also possible to distribute new configuration to all endpoint groups using the same configuration schema in one go



Configuring

This section provides guidance on how to create a configuration schema in Kaa and use it for loading data into applications.

Configuration schema

The structure of the data in Kaa is determined by the configuration schema which is created in the Avro format. See the Configuration section in the Design reference for more information on the schema format.

In this guide, we are going to use the following configuration schema for our example scenario.

This schema allows you to configure a global log level (the **global_log_level** field) for your project and an individual log level per each project module.

```
{
   "type":"record",
   "name":"LogConfiguration",
   "namespace":"org.kaaproject.kaa.schema.sample",
   "fields":[
      {
         "name":"global_log_level",
         "type":{
            "type":"enum",
            "name":"LogLevel",
            "namespace":"org.kaaproject.kaa.schema.sample",
            "symbols":[
               "FATAL",
               "ERROR",
               "WARNING",
               "INFO",
               "DEBUG",
               "TRACE"
            ]
         }
      },
      {
         "name":"modules",
         "type":{
            "type":"array",
            "items":{
               "type":"record",
               "name":"ModuleLogLevel",
               "namespace":"org.kaaproject.kaa.schema.sample",
               "fields":[
                  {
                     "name":"module_name",
                     "type":"string"
                  },
                  {
                     "name":"log_level",
                     "type":"org.kaaproject.kaa.schema.sample.LogLevel"
                  }
               ]
            }
         }
      }
   ]
}
```

Upload configuration schema using UI/REST

Refer to Admin UI or REST API guides for instructions on uploading the configuration schema to the server.

Base and override schemas

After the configuration schema has been successfully uploaded, Kaa generates a *base* and an *override* schemas as described in the Configuration section of the Design reference.

The main difference between these two schemas is that using an override schema you can set some fields as *unchanged* rather than assigning them specific values. For the unchanged field, its value will be taken from the corresponding field in the All group, if it's possible. If not (for example, if there's a new array item which is not present in the All group), Kaa will generate the default value for the unchanged field as described in the Configuration section of the Design reference.

You can obtain all these configuration schemas using REST API (alternatively, use Avro UI forms in Admin UI to work with schemas).

The base schema generated from the configuration schema in our example looks as follows:

```
{
   "type":"record",
   "name":"LogConfiguration",
   "namespace":"org.kaaproject.kaa.schema.sample",
   "fields":[
      {
         "name":"global_log_level",
         "type":{
            "type":"enum",
            "name":"LogLevel",
            "namespace":"org.kaaproject.kaa.schema.sample",
            "symbols":[
               "FATAL",
               "ERROR",
               "WARNING",
               "INFO",
               "DEBUG",
               "TRACE"
            ]
         }
      },
      {
         "name":"modules",
         "type":{
            "type":"array",
            "items":{
               "type":"record",
               "name":"ModuleLogLevel",
               "namespace":"org.kaaproject.kaa.schema.sample",
               "fields":[
                  {
                     "name":"module_name",
                     "type":"string"
                  },
                  {
                     "name":"log_level",
                     "type":"org.kaaproject.kaa.schema.sample.LogLevel"
                  },
                  {
                     "name":"__uuid",
                     "type":[
                        {
                           "name":"uuidT",
                           "type":"fixed",
                           "size":16,
                           "namespace":"org.kaaproject.configuration"
                        },
                        "null"
                     ]
                  }
               ]
            }
         }
      },
      {
         "name":"__uuid",
         "type":[
            "org.kaaproject.configuration.uuidT",
            "null"
         ]
      }
   ]
}
```

The override schema generated from the configuration schema in our example looks as follows:

```
{
   "type":"record",
```

```
"name":"LogConfiguration",
"namespace":"org.kaaproject.kaa.schema.sample",
"fields":[
   {
      "name":"global_log_level",
      "type":[
         {
            "type":"enum",
            "name":"LogLevel",
            "namespace":"org.kaaproject.kaa.schema.sample",
            "symbols":[
               "FATAL",
               "ERROR",
               "WARNING",
               "INFO",
               "DEBUG",
               "TRACE"
            ]
         },
         {
            "symbols":[
               "unchanged"
            ],
            "name":"unchangedT",
            "type":"enum",
            "namespace":"org.kaaproject.configuration"
         }
      ]
   },
   {
      "name":"modules",
      "type":[
         {
            "type":"array",
            "items":{
               "type":"record",
               "name":"ModuleLogLevel",
               "namespace":"org.kaaproject.kaa.schema.sample",
               "fields":[
                  {
                     "name":"module_name",
                     "type":[
                        "string",
                        "org.kaaproject.configuration.unchangedT"
                     ]
                  },
                  {
                     "name":"log_level",
                     "type":[
                        "org.kaaproject.kaa.schema.sample.LogLevel",
                        "org.kaaproject.configuration.unchangedT"
                     ]
                  },
                  {
                     "name":"__uuid",
                     "type":[
                        {
                           "name":"uuidT",
                           "type":"fixed",
                           "size":16,
                           "namespace":"org.kaaproject.configuration"
                        },
                        "null"
                     ]
                  }
               ]
            }
         },
         "org.kaaproject.configuration.unchangedT"
      ]
   },
```

```
      {
         "name":"__uuid",
         "type":[
            "org.kaaproject.configuration.uuidT",
            "null"
         ]
      }
   ]
}
```

Load default data into All group

As soon as both schemas are generated, Kaa loads the default data into the All group using the base schema. The default data is constructed according to the field types as described in the Configuration section of the Design reference. In our example, the generated data looks as follows:

```
{
   "global_log_level":"FATAL",
   "modules":[

   ],
   "__uuid":{
      "org.kaaproject.configuration.uuidT":"[6\u0001°SJ<¼(\u000Fõé\u0015\u0018"
   }
}
```

Edit configuration data for All group

Use the base schema when you edit the configuration data for the All group.

To edit the configuration data for the All group, proceed as follows:

1. Obtain the existing configuration data using the Admin UI (select the All endpoint group under the application, then select the endpoint group configuration) or REST API.
2. In our example, let's set the **global_log_level** field to "TRACE" .
   This global log level will become default for all endpoints.

   IMPORTANT
   The **__uuid** field is reserved for Kaa needs. Never edit its value by yourself.

   The resulting configuration data looks as follows:

   ```
   {
      "global_log_level":"TRACE",
      "modules":[

      ],
      "__uuid":{
         "org.kaaproject.configuration.uuidT":"[6\u0001°SJ<¼(\u000Fõé\u0015\u0018"
      }
   }
   ```

3. To save the changes, click **Save** in Admin UI or use REST API.
4. To apply the saved changes to endpoints, activate the new configuration by clicking **Activate** in Admin UI or using REST API.

Load configuration data into specific group

To create an individual configuration for a specific endpoint group, use the override schema.

In the following configuration example for the *Android Froyo endpoints* group, we've set the **global_log_level** field to "unchanged" (that is, the default value from the All group will be used) and assigned the "DEBUG" level to the **org.kaaproject.kaa.schema.sample.FroyoSpecificModule** module.

```
{
   "global_log_level":{
      "org.kaaproject.configuration.unchangedT":"unchanged"
   },
   "modules":{
      "array":[
         {
            "module_name":{
               "string":"org.kaaproject.kaa.schema.sample.FroyoSpecificModule"
            },
            "log_level":{
               "org.kaaproject.kaa.schema.sample.LogLevel":"DEBUG"
            },
            "__uuid":null
         }
      ]
   },
   "__uuid":null
}
```

In the following configuration example for the *Android endpoints* group, we've assigned the 'INFO" level to the **org.kaaproject.kaa.schema.sample.AndroidModule** module.

```
{
   "global_log_level":{
      "org.kaaproject.configuration.unchangedT":"unchanged"
   },
   "modules":{
      "array":[
         {
            "module_name":{
               "string":"org.kaaproject.kaa.schema.sample.AndroidModule"
            },
            "log_level":{
               "org.kaaproject.kaa.schema.sample.LogLevel":"INFO"
            },
            "__uuid":null
         }
      ]
   },
   "__uuid":null
}
```

Please note that we've left the null value for all __**uuid** fields.

To upload a new configuration for each group, use Admin UI or REST API. Don't forget to activate each configuration as described in the *Edit configuration data for All group* section.

Edit configuration data for specific group

Editing process for a specific group is similar to editing for the All group described in the *Edit configuration data for All group* section.

1. Obtain the current configuration data using the Admin UI (select the endpoint group under the application, then select the endpoint group configuration) or REST API.
   For the *Android Froyo endpoints* group, the result is presented in the following code block. It includes the data we've added in the *Load configuration data into specific group* section, as well as the __**uuid** values automatically generated by Kaa after we've uploaded our configuration to the server.

   NOTE
   Actual __**uuid** values may differ from those provided in this sample.

```
{
   "global_log_level":{
      "org.kaaproject.configuration.unchangedT":"unchanged"
   },
   "modules":{
      "array":[
         {
            "module_name":{
               "string":"org.kaaproject.kaa.schema.sample.FroyoSpecificModule"
            },
            "log_level":{
               "org.kaaproject.kaa.schema.sample.LogLevel":"DEBUG"
            },
            "__uuid":{
               "org.kaaproject.configuration.uuidT":"=ºÞNEµ½ ¯]\"Ê\u001A"
            }
         }
      ]
   },
   "__uuid":{
      "org.kaaproject.configuration.uuidT":"×dëîCrG®auòë\u001D\u0017"
   }
}
```


2. In our example, let's assign the "WARNING" value to the **org.kaaproject.kaa.schema.sample.FroyoSpecificModule** module.


```
{
   "global_log_level":{
      "org.kaaproject.configuration.unchangedT":"unchanged"
   },
   "modules":{
      "array":[
         {
            "module_name":{
               "string":"org.kaaproject.kaa.schema.sample.FroyoSpecificModule"
            },
            "log_level":{
               "org.kaaproject.kaa.schema.sample.LogLevel":"WARNING"
            },
            "__uuid":{
               "org.kaaproject.configuration.uuidT":"=ºÞNEµ½ ¯]\"Ê\u001A"
            }
         }
      ]
   },
   "__uuid":{
      "org.kaaproject.configuration.uuidT":"×dëîCrG®auòë\u001D\u0017"
   }
}
```


3. Load this data back to the server and activate the new configuration.

```
{
  "global_log_level":{
     "org.kaaproject.configuration.unchangedT":"unchanged"
  },
  "modules":{
     "array":[
        {
           "module_name":{
              "string":"org.kaaproject.kaa.schema.sample.FroyoSpecificModule"
           },
           "log_level":{
              "org.kaaproject.kaa.schema.sample.LogLevel":"WARNING"
           },
           "__uuid":{
              "org.kaaproject.configuration.uuidT":"=ºÞNEµ½ ¯]\"Ê\u001A"
           }
        }
     ]
  },
  "__uuid":{
     "org.kaaproject.configuration.uuidT":"×dëîCrG®auòë\u001D\u0017"
  }
}
```

Coding

This section provides code samples which illustrate how to use Kaa SDK to work with configuration schema/data in Kaa, for instance, how to subscribe to configuration updates and implement configuration schema persistence.

Configuration traversal

You can upload the current configuration data to the client and use this data for specific purposes.

> **NOTE**
> The following example for Java illustrates how to upload the configuration data to the client, use it for retrieving log levels of each project module, and check whether a specific module is loggable against the provided log level.

Subscribe to configuration updates

You can subscribe your own callback on any configuration update, as illustrated by the following code example.

In the following example, the argument of the callback contains the full latest configuration data.

Configuration data persistence

During its work, the Kaa client can receive configuration updates. By default, the configuration is not persisted by the client, except for the C SDK (for POSIX build the "kaa_configuration.bin" file will be used).

The following code example illustrates how to use the default implementation of configuration storage.

Also, you have the ability to create your own mechanism of configuration persistence.

The following code example illustrates how to persist configuration data by storing it to a file with the custom storage implementation. To provide the custom storage implementation for the C SDK, just remove the existing implementation from the SDK build (i.e.

posix_configuration_persistence.c).
