Creating custom log appender

To implement a custom log appender, you need to complete the following steps.

1. Design and compile a configuration schema.
2. Implement the log appender based on AbstractLogAppender.
3. Develop the log appender descriptor.
4. Provision the log appender.

We recommend that you use one of the existing log appender implementations as a reference.

Configuration schema

A log appender configuration schema is an Avro compatible schema that defines configuration parameters of the log appender. The following parameters in the schema affect Kaa Admin UI layout.

- minRowCount - specifies a minimum number of rows in a UI table (only for arrays in the schema)
- displayName - displays the name of the field on UI
- displayNames - displays the name of each enumeration symbol on UI (only for enumeration fields in the schema)
- default - displays the default value of the field on UI
- optional - defines whether the field on UI is optional or mandatory
- weight - defines a relative width of the corresponding column on UI (only for arrays in the schema)

```
{
    "namespace":"org.kaaproject.kaa.schema.sample",
    "type":"record",
    "name":"CustomAppenderConfiguration",
    "fields":[
        {
            "name":"servers",
            "displayName":"Your server list",
            "minRowCount":1,
            "type":{
                "type":"array",
                "items":{
                    "namespace":"com.company.kaa.appender",
                    "type":"record",
                    "name":"Server",
                    "fields":[
                        {
                            "name":"host",
                            "displayName":"Host",
                            "weight":0.75,
                            "default":"localhost",
                            "type":"string"
                        },
                        {
                            "name":"port",
                            "displayName":"Port",
                            "weight":0.25,
                            "default":80,
                            "type":"int"
                        }
                    ]
                }
            }
        },
        {
            "name":"StringParameter",
            "displayName":"String parameter name",
            "type":"string"
        },
        {
            "name":"IntegerParameter",
            "displayName":"Integer parameter name",
            "default":1,
            "optional":true,
            "type":[
                "int",
                "null"
            ]
        }
    ]
}
```

Once you have prepared your schema, you can compile it using the following command.
`java -jar /path/to/avro-tools-1.7.7.jar compile schema <schema file> <destination>`
Please refer to Compiling the schema for more information. It is also possible to integrate the schema compilation with avro-maven-plugin.

Log appender implementation

All Kaa log appenders extend generic abstract class org.kaaproject.kaa.server.common.log.shared.appender.AbstractLogAppender<T>. The following code example illustrates the implementation of a custom log appender.

```
package org.kaaproject.kaa.sample.appender;

import org.kaaproject.kaa.schema.sample.CustomAppenderConfiguration;
import org.kaaproject.kaa.common.dto.logs.LogAppenderDto;
import org.kaaproject.kaa.common.dto.logs.LogEventDto;

/**
 *
 * Sample appender implementation that uses {@link CustomAppenderConfiguration} as configuration.
 *
 */
public class CustomLogAppender extends AbstractLogAppender<CustomAppenderConfiguration> {

    /**
     * Inits the appender from configuration.
     *
     * @param appender the metadata object that contains useful info like application token, tenant id, etc.
     * @param configuration the configuration object that you have specified during appender provisioning.
     */
    @Override
    protected void initFromConfiguration(LogAppenderDto appender, CustomAppenderConfiguration configuration)
{
                //Do some initialization here.
    }

    /**
     * Consumes and delivers logs.
     *
     * @param logEventPack container for log events with some metadata like log event schema.
     * @param header additional data about log event source (endpoint key hash, application token, header
version, timestamp).
     */
    @Override
    public void doAppend(LogEventPack logEventPack, RecordHeader header) {
                //Append logs to your system here.
    }

    /**
     * Closes this appender and releases any resources associated with it.
     *
     */
    @Override
    public void close() {
                //Free allocated resources here.
    }
}
```

Log appender descriptor

A log appender descriptior provides Kaa with the information on how to locate and configure your custom log appender. To implement a log appender descriptor, you need to implement the AppenderConfig interface at first.

It is also important to provide your class with the @KaaPluginConfig annotiation. This annotation helps Kaa Admin UI to find all available log appenders in the class path.

> **NOTE**
> A log appender descriptor is optional if you are going to configure your log appenders using only REST API.

The following code example illustrates the implementation of a log appender descriptor.

```
package org.kaaproject.kaa.sample.appender.config;

import org.apache.avro.Schema;
import org.kaaproject.kaa.schema.sample.CustomAppenderConfiguration;
import org.kaaproject.kaa.server.common.log.shared.annotation.KaaAppenderConfig;
import org.kaaproject.kaa.server.common.log.shared.config.AppenderConfig;

/**
 *
 * Sample descriptor for org.kaaproject.kaa.sample.appender.CustomLogAppender appender.
 *
 */
@KaaPluginConfig(pluginType = PluginType.LOG_APPENDER)
public class CustomAppenderDescriptor implements AppenderConfig {


    public CustomAppenderDescriptor() {
    }
    /**
     * Name of the appender will be used in Admin UI
     */
    @Override
    public String getName() {
        return "Custom appender";
    }
    /**
     * Returns name of the appender class.
     */
    @Override
    public String getLogAppenderClass() {
        return "org.kaaproject.kaa.sample.appender.CustomLogAppender";
    }
    /**
     * Returns avro schema of the appender configuration.
     */
    @Override
    public Schema getConfigSchema() {
        return CustomAppenderConfiguration.getClassSchema();
    }
}
```

Log appender provisioning

To provision your log appender, do the following:

1. Place the log appender descriptor and configuration classes into the Admin UI class path.
2. Place the log appender implementation classes into the Operations Server class path.
3. Use Admin UI or REST API to create/update/delete your log appender instances.