Log appenders

A log appender is a service utility which resides on the Operations server. This utility is responsible for writing logs received by the Operations server to a single specific storage, as defined by the log appender's type. Each Kaa application may use only one log appender at a time. A Kaa developer is able to add, update and delete log appenders using Admin UI or REST API. Kaa provides several default implementations of log appenders. It is also possible to create custom log appenders.

Default log appenders

There are several log appender implementations that are available out of the box for each Kaa installation. This page contains general information about log appender architecture, configuration and administration.

File system log appender

Architecture

The file system log appender stores received logs into the local file system of the Operations server. This log appender may be used for test purposes or in pair with tools like Flume and others. Logs are stored in files under the `/$logsRootPath/tenant_$tenantId/application_$applicationId` folder, where `logsRootPath` is a configuration parameter, `tenantId` and `applicationId` are ids of the current tenant and application respectively. Access to the logs is controlled via Linux file system permissions.

You can log in to the Operations server host and browse logs using the *kaa_log_user_$applicationToken* user name and pubic key, which is created as a part of the configuration.

Configuration

The file system log appender configuration should match the following Avro schema.

```
{
    "namespace":"org.kaaproject.kaa.server.appenders.file.config.gen",
    "type":"record",
    "name":"FileConfig",
    "fields":[
        {
            "name":"publicKey",
            "displayName":"Public Key",
            "maxLength":1000,
            "default":"",
            "type":"string"
        },
        {
            "name":"logsRootPath",
            "displayName":"Logs root path",
            "default":"/kaa_log_uploads",
            "type":"string"
        },
        {
            "name":"rollingFileNamePatern",
            "displayName":"Rolling file name pattern",
            "default":"logFile.%d{yyyy-MM-dd}.log",
            "type":"string"
        },
        {
            "name":"rollingMaxHistory",
            "displayName":"Rolling max history",
            "default":30,
            "type":"int"
        },
        {
            "name":"triggerMaxFileSize",
            "displayName":"Trigger max file size",
            "default":"1GB",
            "type":"string"
        },
        {
            "name":"encoderPattern",
            "displayName":"Encoder pattern",
            "default":"%-4relative [%thread] %-5level %logger{35} - %msg%n",
            "type":"string"
        }
    ]
}
```

The following configuration example matches the previous schema.

```
{
    "publicKey":"XXXXXXXXXXXXXXXXXXXXXXXXXX",
    "logsRootPath":"/kaa_log_uploads",
    "rollingFileNamePatern":"logFile.%d{yyyy-MM-dd}.log",
    "rollingMaxHistory":30,
    "triggerMaxFileSize":"1GB",
    "encoderPattern":"%-4relative [%thread] %-5level %logger{35} - %msg%n"
}
```

Administration

To create the file system log appender, use either Admin UI or REST API. The following REST API call example illustrates how to create a new file system log appender.

```
curl -v -S -u devuser:devuser123 -X POST -H 'Content-Type: application/json' -d'{"appenderClassName": "org.
kaaproject.kaa.server.appenders.file.appender.FileSystemLogAppender", "applicationId": "70",
"applicationToken": "946558468095768", "configuration": "{\"publicKey\":\"XXXXXXXXXXXXXXXXXXXXXXXXXX\",\"
logsRootPath\":\"/kaa_log_uploads\",\"rollingFileNamePatern\":\"logFile.%d{yyyy-MM-dd}.log\",\"
rollingMaxHistory\":30,\"triggerMaxFileSize\":\"1GB\",\"encoderPattern\":\"%-4relative [%thread] %-5level %
logger{35} - %msg%n\"}", "description": "New sample file system log appender", "headerStructure": [ "KEYHASH","
TIMESTAMP" ], "name": "New file system appender", "schema": { "id": "92", "majorVersion": 1, "minorVersion": 0
}, "status": "REGISTERED", "tenantId": "10", "typeName": "File"}' "http://localhost:8080/kaaAdmin/rest/api
/logAppender" | python -mjson.tool
```

```
{
    "appenderClassName": "org.kaaproject.kaa.server.appenders.file.appender.FileSystemLogAppender",
    "applicationId": "70",
    "applicationToken": "946558468095768",
    "configuration": "{\"publicKey\":\"XXXXXXXXXXXXXXXXXXXXXXXXXX\",\"logsRootPath\":\"
/kaa_log_uploads_new\",\"rollingFileNamePatern\":\"logFile.%d{yyyy-MM-dd}.log\",\"rollingMaxHistory\":30,\"
triggerMaxFileSize\":\"1GB\",\"encoderPattern\":\"%-4relative [%thread] %-5level %logger{35} - %msg%n\"}",
    "createdTime": 1417006362287,
    "createdUsername": "devuser",
    "description": "New sample file system log appender",
    "headerStructure": [
        "KEYHASH",
        "TIMESTAMP"
    ],
    "id": "161",
    "name": "New file system appender",
    "schema": {
        "id": "92",
        "majorVersion": 1,
        "minorVersion": 0
    },
    "schemaVersion": "1.0",
    "status": "REGISTERED",
    "tenantId": "10",
    "typeName": "File"
}
```

Flume log appender

Architecture

The Flume log appender encapsulates received logs into a Flume event and sends this event to external Flume sources via Avro RPC. Log records are serialized by the following RecordData schema as a binary avro file and stored in the Flume event raw body.

The RecordData schema has the following four fields.

- recordHeader
- schemaVersion
- applicationToken
- eventRecords

The **recordHeader** field stores a set of log metadata fields.

The **eventRecords** field stores an array of raw records. Each element of the array is a log record in the Avro binary format serialized by the log schema.

The **schemaVersion** and **applicationToken** fields should be used as parameters of a REST API call to Kaa in order to obtain the logs Avro schema for **eventRecords** and enable parsing of the binary data.

```
{
    "type":"record",
    "name":"RecordData",
    "namespace":"org.kaaproject.kaa.server.common.log.shared.avro.gen",
    "fields":[
        {
            "name":"recordHeader",
            "type":[
                {
```

```
                    "type":"record",
                    "name":"RecordHeader",
                    "namespace":"org.kaaproject.kaa.server.common.log.shared.avro.gen",
                    "fields":[
                        {
                            "name":"endpointKeyHash",
                            "type":[
                                {
                                    "type":"string"
                                },
                                "null"
                            ]
                        },
                        {
                            "name":"applicationToken",
                            "type":[
                                {
                                    "type":"string"
                                },
                                "null"
                            ]
                        },
                        {
                            "name":"headerVersion",
                            "type":[
                                {
                                    "type":"int"
                                },
                                "null"
                            ]
                        },
                        {
                            "name":"timestamp",
                            "type":[
                                {
                                    "type":"long"
                                },
                                "null"
                            ]
                        }
                    ]
                },
                "null"
            ]
        },
        {
            "name":"schemaVersion",
            "type":"int"
        },
        {
            "name":"applicationToken",
            "type":"string"
        },
        {
            "name":"eventRecords",
            "type":{
                "type":"array",
                "items":{
                    "name":"RecordEvent",
                    "namespace":"org.kaaproject.kaa.server.common.log.shared.avro.gen",
                    "type":"bytes"
                }
            }
        }
    ]
}
```

In addition, Kaa provides the following two extended Flume agents which can be used together with the Flume log appender.

- Kaa flume source

- Kaa flume sink

The Kaa flume source is a Flume agent with an extension to the standard Flume NG Avro Sink that includes additional features and performance improvements. The Kaa flume source receives data from the Flume log appender and delivers it to an external Avro Source located in the Hadoop cluster.

The Kaa flume sink is a Flume agent with an extension to the standard Flume NG HDFS Sink that includes additional features. The Kaa flume sink is aware of the log records data schema and stores log data into HDFS as Avro Sequence files using the following Avro schema.

```
{
   "type":"record",
   "name":"RecordWrapper",
   "namespace":"org.kaaproject.kaa.log",
   "fields":[
      {
         "name":"recordHeader",
         "type":[
            {
               "type":"record",
               "name":"RecordHeader",
               "namespace":"org.kaaproject.kaa.server.common.log.shared.avro.gen",
               "fields":[
                  {
                     "name":"endpointKeyHash",
                     "type":[
                        {
                           "type":"string"
                        },
                        "null"
                     ]
                  },
                  {
                     "name":"applicationToken",
                     "type":[
                        {
                           "type":"string"
                        },
                        "null"
                     ]
                  },
                  {
                     "name":"headerVersion",
                     "type":[
                        {
                           "type":"int"
                        },
                        "null"
                     ]
                  },
                  {
                     "name":"timestamp",
                     "type":[
                        {
                           "type":"long"
                        },
                        "null"
                     ]
                  }
               ]
            },
            "null"
         ]
      },
      {
         "name":"recordData",
         "type":[
            ${record_data_schema},
            "null"
         ]
      }
   ]
}
```

${record_data_schema} - is a variable which is substituted at run time by Kaa HDFS Sink with the actual logs Avro schema. The actual logs Avro schema is obtained via a REST API call to Kaa.

Configuration

A Flume appender configuration should match the following Avro schema.

```
{
   "namespace":"org.kaaproject.kaa.server.appenders.flume.config.gen",
   "type":"record",
   "name":"FlumeConfig",
   "fields":[
      {
         "name":"hostsBalancing",
         "displayName":"Hosts balancing",
         "type":[
            {
               "type":"record",
               "namespace":"org.kaaproject.kaa.server.appenders.flume.config.gen",
               "name":"PrioritizedFlumeNodes",
               "displayName":"Prioritized",
               "fields":[
                  {
                     "name":"flumeNodes",
                     "displayName":"Flume nodes",
                     "minRowCount":1,
                     "type":{
                        "type":"array",
                        "items":{
                           "namespace":"org.kaaproject.kaa.server.appenders.flume.config.gen",
                           "type":"record",
                           "name":"PrioritizedFlumeNode",
                           "fields":[
                              {
                                 "name":"host",
                                 "displayName":"Host",
                                 "weight":0.8,
                                 "default":"localhost",
                                 "type":"string"
                              },
                              {
                                 "name":"port",
                                 "displayName":"Port",
                                 "weight":0.1,
                                 "default":7070,
                                 "type":"int"
                              },
                              {
                                 "name":"priority",
                                 "displayName":"Priority",
                                 "weight":0.1,
                                 "default":1,
                                 "type":"int"
                              }
                           ]
                        }
                     }
                  }
               ]
            },
            {
               "type":"record",
               "namespace":"org.kaaproject.kaa.server.appenders.flume.config.gen",
               "name":"FlumeNodes",
               "displayName":"Round Robin",
               "fields":[
                  {
                     "name":"flumeNodes",
                     "displayName":"Flume nodes",
                     "minRowCount":2,
                     "type":{
                        "type":"array",
                        "items":{
                           "namespace":"org.kaaproject.kaa.server.appenders.flume.config.gen",
                           "type":"record",
```

```
                        "name":"FlumeNode",
                        "fields":[
                           {
                              "name":"host",
                              "displayName":"Host",
                              "weight":0.75,
                              "default":"localhost",
                              "type":"string"
                           },
                           {
                              "name":"port",
                              "displayName":"Port",
                              "weight":0.25,
                              "default":7070,
                              "type":"int"
                           }
                        ]
                     }
                  }
               }
            ]
         }
      ]
   }
}
```

The following configuration example matches the previous schema.

```
{
    "hostsBalancing":{
        "org.kaaproject.kaa.server.appenders.flume.config.gen.PrioritizedFlumeNodes":{
            "flumeNodes":[
                {
                    "host":"10.2.1.55",
                    "port":7070,
                    "priority":1
                }
            ]
        }
    }
}
```

NOTE

Flume log appenders can have either *prioritized* or *round robin* host balancing.
For the prioritized host balancing, every flume node record should have a host address, port and priority. The highest priority is *1*. When choosing a server to which to save logs, an endpoint will send requests to the servers starting from the server with the highest priority.
For the round robin host balancing, every flume node record should have a host address and port. When choosing a server to which to save logs, an endpoint will send requests to the servers according to the round robin algorithm.

Administration

To create a flume log appender, use either Admin UI or REST API. The following REST API call example illustrates how to create a new flume log appender.

```
curl -v -S -u devuser:devuser123 -X POST -H 'Content-Type: application/json' -d'{"appenderClassName": "org.
kaaproject.kaa.server.appenders.flume.appender.FlumeLogAppender", "applicationId": "70", "applicationToken":
"946558468095768", "configuration": "{\"hostsBalancing\":{\"org.kaaproject.kaa.server.appenders.flume.config.
gen.PrioritizedFlumeNodes\":{\"flumeNodes\":[{\"host\":\"10.2.1.55\",\"port\":7070,\"priority\":1}]}}}",
"description": "Sample Flume log appender", "headerStructure": [ "KEYHASH","TIMESTAMP" ], "name": "Flume
appender", "schema": { "id": "92", "majorVersion": 1, "minorVersion": 0 }, "status": "REGISTERED", "tenantId":
"10", "typeName": "Flume"}' "http://localhost:8080/kaaAdmin/rest/api/logAppender" | python -mjson.tool
```

```
{
    "appenderClassName": "org.kaaproject.kaa.server.appenders.flume.appender.FlumeLogAppender",
    "applicationId": "70",
    "applicationToken": "946558468095768",
    "configuration": "{\"hostsBalancing\":{\"org.kaaproject.kaa.server.appenders.flume.config.gen.
PrioritizedFlumeNodes\":{\"flumeNodes\":[{\"host\":\"10.2.1.55\",\"port\":7070,\"priority\":1}]}}}",
    "createdTime": 1417090601467,
    "createdUsername": "devuser",
    "description": "Sample Flume log appender",
    "headerStructure": [
        "KEYHASH",
        "TIMESTAMP"
    ],
    "id": "163",
    "name": "Flume appender",
    "schema": {
        "id": "92",
        "majorVersion": 1,
        "minorVersion": 0
    },
    "schemaVersion": "1.0",
    "status": "REGISTERED",
    "tenantId": "10",
    "typeName": "Flume"
}
```

If you want to use Flume agents together with the Flume log appender, create necessary Kaa Flume agents as described in Installing Kaa flume agents.

MongoDB log appender

Architecture

The MongoDB log appender is responsible for transferring logs from the Operations server to the MongoDB database. The logs are stored in the table named *logs_$applicationToken*, where *$applicationToken* matches the token of the current application.

Configuration

The MongoDB log appender configuration should match the following Avro schema.

```
{
    "namespace":"org.kaaproject.kaa.server.appenders.mongo.config.gen",
    "type":"record",
    "name":"MongoDbConfig",
    "fields":[
        {
            "name":"mongoServers",
            "displayName":"MongoDB nodes",
            "minRowCount":1,
            "type":{
                "type":"array",
                "items":{
                    "namespace":"org.kaaproject.kaa.server.appenders.mongo.config.gen",
                    "type":"record",
                    "name":"MongoDbServer",
                    "fields":[
                        {
                            "name":"host",
                            "displayName":"Host",
                            "weight":0.75,
                            "default":"localhost",
                            "type":"string"
                        },
                        {
                            "name":"port",
                            "displayName":"Port",
                            "weight":0.25,
                            "default":27017,
```

```
                    "type":"int"
                }
            ]
        }
    }
},
{
    "name":"mongoCredentials",
    "displayName":"Authentication credentials",
    "minRowCount":0,
    "type":{
        "type":"array",
        "items":{
            "namespace":"org.kaaproject.kaa.server.appenders.mongo.config.gen",
            "type":"record",
            "name":"MongoDBCredential",
            "fields":[
                {
                    "name":"user",
                    "displayName":"User",
                    "weight":0.5,
                    "default":"user",
                    "type":"string"
                },
                {
                    "name":"password",
                    "displayName":"Password",
                    "weight":0.5,
                    "default":"password",
                    "type":"string"
                }
            ]
        }
    }
},
{
    "name":"dbName",
    "displayName":"MongoDB database name",
    "type":"string"
},
{
    "name":"connectionsPerHost",
    "displayName":"Max connections per host",
    "default":30,
    "optional":true,
    "type":[
        "int",
        "null"
    ]
},
{
    "name":"maxWaitTime",
    "displayName":"Max wait time (ms)",
    "default":120000,
    "optional":true,
    "type":[
        "int",
        "null"
    ]
},
{
    "name":"connectionTimeout",
    "displayName":"Connection timeout (ms)",
    "default":5000,
    "optional":true,
    "type":[
        "int",
        "null"
    ]
},
{
```

```
            "name":"socketTimeout",
            "displayName":"Socket timeout (ms)",
            "default":0,
            "optional":true,
            "type":[
                "int",
                "null"
            ]
        },
        {
            "name":"socketKeepalive",
            "displayName":"Turn on socket keepalive",
            "default":false,
            "optional":true,
            "type":[
                "boolean",
                "null"
            ]
        },
        {
            "name":"autoConnectRetry",
            "displayName":"Automatic reconnect on errors",
            "default":true,
            "optional":true,
            "type":[
                "boolean",
                "null"
            ]
        }
    ]
}
```

The following configuration example matches the previous schema.

```
{
    "mongoServers":[
        {
            "host":"127.0.0.1",
            "port":27001
        }
    ],
    "mongoCredentials":[
        {
            "user":"your_user_name",
            "password":"your_password"
        }
    ],
    "dbName":"test_db",
    "connectionsPerHost":{"int":10},
    "maxWaitTime":{"int":120000},
    "connectionTimeout":{"int":5000},
    "socketTimeout":{"int":60},
    "socketKeepalive":{"boolean":true},
    "autoConnectRetry":{"boolean":true}
}
```

Administration

To create the MongoDB log appender, use Admin UI or REST API. The following REST API call example illustrates how to create a new MongoDB log appender.

```
curl -v -S -u devuser:devuser123 -X POST -H 'Content-Type: application/json' -d'{"appenderClassName": "org.
kaaproject.kaa.server.appenders.mongo.appender.MongoDbLogAppender", "applicationId": "70", "applicationToken":
"946558468095768", "configuration": "{\"mongoServers\":[{\"host\":\"127.0.0.1\",\"port\":27017}],\"
mongoCredentials\":[{\"user\":\"your_user_name\",\"password\":\"your_user_password\"}],\"dbName\":\"test_db\",\"
connectionsPerHost\":{\"int\":30},\"maxWaitTime\":{\"int\":120000},\"connectionTimeout\":{\"int\":5000},\"
socketTimeout\":{\"int\":0},\"socketKeepalive\":{\"boolean\":false},\"autoConnectRetry\":{\"boolean\":true}}",
"description": "New sample Mongo db log appender", "headerStructure": [ "KEYHASH","TIMESTAMP" ], "name": "New
Mongo DB appender", "schema": { "id": "92", "majorVersion": 1, "minorVersion": 0 }, "status": "REGISTERED",
"tenantId": "10", "typeName": "File"}' "http://localhost:8080/kaaAdmin/rest/api/logAppender" | python -mjson.
tool
```

```
{
    "appenderClassName":"org.kaaproject.kaa.server.appenders.mongo.appender.MongoDbLogAppender",
    "applicationId":"70",
    "applicationToken":"946558468095768",
    "configuration":"{\"mongoServers\":[{\"host\":\"127.0.0.1\",\"port\":27017}],\"dbName\":\"test_db\",\"
connectionsPerHost\":{\"int\":30},\"maxWaitTime\":{\"int\":120000},\"connectionTimeout\":{\"int\":5000},\"
socketTimeout\":{\"int\":0},\"socketKeepalive\":{\"boolean\":false},\"autoConnectRetry\":{\"boolean\":
true}}",
    "createdTime":1417105170741,
    "createdUsername":"devuser",
    "description":"Sample MongoDB log appender",
    "headerStructure":[
        "KEYHASH",
        "TOKEN"
    ],
    "id":"164",
    "name":"MongoDB appender",
    "schema":{
        "id":"92",
        "majorVersion":1,
        "minorVersion":0
    },
    "schemaVersion":"1.0",
    "status":"REGISTERED",
    "tenantId":"10",
    "typeName":"Mongo"
}
```

CDAP log appender

Architecture

The CDAP log appender is responsible for the logs transfer to the CDAP platform. Logs are stored in a stream that is specified by the *stream* configuration parameter.

Configuration

The CDAP log appender configuration should match the following Avro schema.

```
{
    "namespace":"org.kaaproject.kaa.server.appenders.cdap.config.gen",
    "type":"record",
    "name":"CdapConfig",
    "fields":[
        {
            "name":"stream",
            "displayName":"Stream",
            "default":"stream",
            "type":"string"
        },
        {
            "name":"host",
            "displayName":"Host",
            "default":"localhost",
```

```
        "type":"string"
    },
    {
        "name":"port",
        "displayName":"Port",
        "default":10000,
        "type":"int"
    },
    {
        "name":"ssl",
        "displayName":"Use SSL",
        "optional":true,
        "type":[
            "boolean",
            "null"
        ]
    },
    {
        "name":"verifySslCert",
        "displayName":"Validate SSL Certificate",
        "optional":true,
        "type":[
            "boolean",
            "null"
        ]
    },
    {
        "name":"writerPoolSize",
        "displayName":"Writer thread/connection pool size",
        "optional":true,
        "type":[
            "int",
            "null"
        ]
    },
    {
        "name":"callbackThreadPoolSize",
        "displayName":"Callback thread pool size",
        "optional":true,
        "default":2,
        "type":[
            "int",
            "null"
        ]
    },
    {
        "name":"version",
        "displayName":"Version",
        "optional":true,
        "type":[
            "string",
            "null"
        ]
    },
    {
        "name":"authClient",
        "displayName":"Authentication client class name",
        "optional":true,
        "type":[
            "string",
            "null"
        ]
    },
    {
        "name":"username",
        "displayName":"Username",
        "optional":true,
        "type":[
            "string",
            "null"
        ]
```

```
        },
        {
            "name":"password",
            "displayName":"Password",
            "optional":true,
            "type":[
                "string",
                "null"
            ]
        }
    ]
}
```

The following configuration example matches the previous schema.

```
{
    "stream":"testStreamName",
    "host":"localhost",
    "port":10000,
    "ssl":null,
    "verifySslCert":null,
    "writerPoolSize":null,
    "callbackThreadPoolSize":{
        "int":2
    },
    "version":null,
    "authClient":null,
    "username":null,
    "password":null
}
```

Administration

To create a CDAP log appender, use either Admin UI or REST API. The following REST API call example illustrates how to create a new CDAP log appender.

```
curl -v -S -u devuser:devuser123 -X POST -H 'Content-Type: application/json' -d'{"appenderClassName": "org.
kaaproject.kaa.server.appenders.cdap.appender.CdapLogAppender", "applicationId": "70", "applicationToken":
"946558468095768", "configuration": "{\"stream\":\"stream\",\"host\":\"localhost\",\"port\":10000,\"ssl\":null,
\"verifySslCert\":null,\"writerPoolSize\":null,\"callbackThreadPoolSize\":{\"int\":2},\"version\":null,\"
authClient\":null,\"username\":null,\"password\":null}", "description": "New sample Mongo db log appender",
"headerStructure": [ "KEYHASH","TIMESTAMP" ], "name": "New Mongo DB appender", "schema": { "id": "92",
"majorVersion": 1, "minorVersion": 0 }, "status": "REGISTERED", "tenantId": "10", "typeName": "File"}'
"http://localhost:8080/kaaAdmin/rest/api/logAppender" | python -mjson.tool
```

```
{
    "appenderClassName":"org.kaaproject.kaa.server.appenders.cdap.appender.CdapLogAppender",
    "applicationId":"70",
    "applicationToken":"946558468095768",
    "configuration":"{\"stream\":\"stream\",\"host\":\"localhost\",\"port\":10000,\"ssl\":null,\"
verifySslCert\":null,\"writerPoolSize\":null,\"callbackThreadPoolSize\":{\"int\":2},\"version\":null,\"
authClient\":null,\"username\":null,\"password\":null}",
    "createdTime":1417105293146,
    "createdUsername":"devuser",
    "description":"Sample Cdap appender",
    "headerStructure":[
        "KEYHASH",
        "TOKEN"
    ],
    "id":"165",
    "name":"Cdap appender",
    "schema":{
        "id":"92",
        "majorVersion":1,
        "minorVersion":0
    },
    "schemaVersion":"1.0",
    "status":"REGISTERED",
    "tenantId":"10",
    "typeName":"Cdap"
}
```

Oracle NoSQL appender

Architecture

The Oracle NoSQL log appender is responsible for transferring logs from the Operations server to the Oracle NoSQL key/value storage. Logs are stored in the key/value storage using the following key path:
${applicationToken}/${logSchemaVersion}/${endpointKeyHash}/${uploadTimestamp}/${counter}

where:

- *applicationToken* - matches the token of the current application
- *logSchemaVersion* - the version of the avro log schema used to serialize log records
- *endpointKeyHash* - a key hash identifying the endpoint which produced the log record
- *uploadTimestamp* - a timestamp in milliseconds when logs were uploaded to the key/value storage
- *counter* - the serial number of the record

Values are stored as serialized GenericRecords using record wrapper avro schema.

Configuration

The Oracle NoSQL log appender configuration should match the following Avro schema.

```
{
   "namespace":"org.kaaproject.kaa.server.appenders.oraclenosql.config.gen",
   "type":"record",
   "name":"OracleNoSqlConfig",
   "fields":[
      {
         "name":"storeName",
         "displayName":"KVStore name",
         "default":"kvstore",
         "type":"string"
      },
      {
         "name":"kvStoreNodes",
         "displayName":"KVStore nodes",
         "minRowCount":1,
         "type":{
            "type":"array",
            "items":{
               "namespace":"org.kaaproject.kaa.server.appenders.oraclenosql.config.gen",
```

```
            "type":"record",
            "name":"KvStoreNode",
            "fields":[
               {
                  "name":"host",
                  "displayName":"Host",
                  "weight":0.75,
                  "default":"localhost",
                  "type":"string"
               },
               {
                  "name":"port",
                  "displayName":"Port",
                  "weight":0.25,
                  "default":5000,
                  "type":"int"
               }
            ]
         }
      }
   },
   {
      "name":"username",
      "displayName":"Username",
      "optional":true,
      "type":[
         "string",
         "null"
      ]
   },
   {
      "name":"walletDir",
      "displayName":"Oracle Wallet directory",
      "optional":true,
      "type":[
         "string",
         "null"
      ]
   },
   {
      "name":"pwdFile",
      "displayName":"Password store file",
      "optional":true,
      "type":[
         "string",
         "null"
      ]
   },
   {
      "name":"securityFile",
      "displayName":"Security properties file",
      "optional":true,
      "type":[
         "string",
         "null"
      ]
   },
   {
      "name":"transport",
      "displayName":"KVStore communication transport",
      "optional":true,
      "type":[
         "string",
         "null"
      ]
   },
   {
      "name":"ssl",
      "displayName":"Security transport",
      "optional":true,
      "type":[
```

```
            "string",
            "null"
         ]
      },
      {
         "name":"sslCipherSuites",
         "displayName":"SSL/TLS cipher suites",
         "optional":true,
         "type":[
            "string",
            "null"
         ]
      },
      {
         "name":"sslProtocols",
         "displayName":"SSL/TLS procotols",
         "optional":true,
         "type":[
            "string",
            "null"
         ]
      },
      {
         "name":"sslHostnameVerifier",
         "displayName":"SSL/TLS hostname verifier",
         "optional":true,
         "type":[
            "string",
            "null"
         ]
      },
      {
         "name":"sslTrustStore",
         "displayName":"Java truststore file location",
         "optional":true,
         "type":[
            "string",
            "null"
         ]
      },
      {
         "name":"sslTrustStoreType",
         "displayName":"Java truststore type",
         "optional":true,
         "type":[
            "string",
            "null"
         ]
      }
   ]
}
```

The following configuration example matches the previous schema.

```
{
   "storeName":"kvstore",
   "kvStoreNodes":[
      {
         "host":"localhost",
         "port":5000
      }
   ],
   "username":null,
   "walletDir":null,
   "pwdFile":null,
   "securityFile":null,
   "transport":null,
   "ssl":null,
   "sslCipherSuites":null,
   "sslProtocols":null,
   "sslHostnameVerifier":null,
   "sslTrustStore":null,
   "sslTrustStoreType":null
}
```

Administration

To create an Oracle NoSQL log appender, use either Admin UI or REST API. The following REST API call example illustrates how to create a new Oracle NoSQL log appender.

```
curl -v -S -u devuser:devuser123 -X POST -H 'Content-Type: application/json' -d'{"appenderClassName": "org.
kaaproject.kaa.server.appenders.oraclenosql.appender.OracleNoSqlLogAppender", "applicationId": "70",
"applicationToken": "946558468095768", "configuration": "{\"storeName\":\"kvstore\",\"kvStoreNodes\":[{\"host\":
\"localhost\",\"port\":5000}],\"username\":null,\"walletDir\":null,\"pwdFile\":null,\"securityFile\":null,\"
transport\":null,\"ssl\":null,\"sslCipherSuites\":null,\"sslProtocols\":null,\"sslHostnameVerifier\":null,\"
sslTrustStore\":null,\"sslTrustStoreType\":null}", "description": "Sample Oracle NoSQL appender",
"headerStructure": [], "name": "Oracle NoSQL appender", "schema": {"id": "92", "majorVersion": 1,
"minorVersion": 0 }, "status": "REGISTERED", "tenantId": "10", "typeName": "Oracle NoSQL"}' "http://localhost:
8080/kaaAdmin/rest/api/logAppender" | python -mjson.tool
```

```
{
    "appenderClassName": "org.kaaproject.kaa.server.appenders.oraclenosql.appender.OracleNoSqlLogAppender",
    "applicationId": "70",
    "applicationToken": "946558468095768",
    "configuration": "{\"storeName\":\"kvstore\",\"kvStoreNodes\":[{\"host\":\"localhost\",\"port\":5000}],\"
username\":null,\"walletDir\":null,\"pwdFile\":null,\"securityFile\":null,\"transport\":null,\"ssl\":null,\"
sslCipherSuites\":null,\"sslProtocols\":null,\"sslHostnameVerifier\":null,\"sslTrustStore\":null,\"
sslTrustStoreType\":null}",
    "createdTime": 1417107992158,
    "createdUsername": "devuser",
    "description": "Sample Oracle NoSQL appender",
    "headerStructure": [],
    "id": "167",
    "name": "Oracle NoSQL appender",
    "schema": {
        "id": "92",
        "majorVersion": 1,
        "minorVersion": 0
    },
    "schemaVersion": "1.0",
    "status": "REGISTERED",
    "tenantId": "10",
    "typeName": "Oracle NoSQL"
}
```

Custom appender implementation

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

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

We recommend that you use one of the existing 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
- 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)

If you are using arrays in your schema, you can specify minimum amount of required elements with "minRowCount" parameter.

Fields in your schema have displayName (display name on the UI), displayNames (display name of each enumeration symbol on the UI), default (default value on the UI), optional and weight (column weight on the UI) parameters.

```
{
    "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.

Appender implementation

All Kaa appenders extend generic abstract class org.kaaproject.kaa.server.common.log.shared.appender.AbstractLogAppender<T>. The following code example illustrates an 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 usefull 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 an appender descriptor, you need to implement the AppenderConfig interface.

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

> **NOTE**
> An appender descriptor is optional if you are going to configure your appenders only through REST API.

The following code example illustrates an 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.
 *
 */
@KaaAppenderConfig
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();
    }
}
```

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 appender instances.
