Olog Service

Release 1.0.0

unknown

Sep 07, 2023

CONTENTS

Olog is an online logbook service which allows for the creation and retrieval of log entries.

The service was developed to address the needs of operators, engineers, and users of large scientific facilities.

Key features:

- Integration with CS-Studio, Phoebus, Bluesky, and other controls and data acquisition tools.
- Tags & Logbooks provide an effective way to organize and sort log entries
- Support for fuzzy searching
- Markup support for creating rich text log entries. Markup is based on the Commonmark specification, extended with support for image size and tables. Clients may request a HTML formatted quick reference (maintained in the project) resource using an URL like http(s)://url.to.service/CommonmarkCheatsheet.html.

CHAPTER

ONE

A LOG ENTRY:


```
{
   "id":575,
   "owner":"testOwner1",
   "source":"**Beam Dump** due to Major power dip Current Alarms Booster transmitter␣
↪switched back to lower state.
            PV : BR-RF{Xmtr-PLC}ICS:Down-Sts",
   "description":"Beam Dump due to Major power dip Current Alarms Booster transmitter␣
↪switched back to lower state.
                PV : BR-RF{Xmtr-PLC}ICS:Down-Sts",
   "level":"Info",
   "title":"Some title",
   "state":"Active",
   "createdDate":1577392617217,
   "logbooks":[
      {
         "name":"ControlsOperations",
         "owner":"operators"
      }
   ],
   "tags":[
      {
         "name":"Fault"
      }
   ],
   "properties":[
      {
         "name":"FaultReport",
         "owner":"testOwner1",
         "attributes":[
            {
               "name":"id",
               "value":"1234"
            },
            {
               "name":"URL",
               "value":"https://nsls2.bnl.gov/faults/1234"
            }
         ]
      }
   ],
```


(continues on next page)

(continued from previous page)


```
   "attachments":[
      {
         "id":"5e21fe829fef8a53f0183d4a",
         "filename":"screenshot_image.png",
         "fileMetadataDescription":"image.png"
      }
   ],
   "events":[
      {
         "name":"faultTime",
         "instant":1577389011004
      }
   ]
}
```


1.1 LogEntry

The most important filed of any log entry are the description and owner.
description: Is the text body of a log entry.
owner: The creator of the log entry.
level Info, Warning,

Other fields include
source: When markup support is enabled the description text + the markup annotations are stored in this field.
id: The unique identifier for a log entry, this is automatically generated at the time of creation.
createTime: Another automatically generated field which stores the unix epoch millisecond at which the log entry was created.

1.2 Attachment

Each log entry can have a list attachments, these can be any type of files.

1.3 Logbooks & Tags

Logbooks and Tags are useful organization tools. Each log entry can be associated with one or more logbooks and have zero or more tags.

1.4 Properties

A property is a list of key value pairs. The provide means of attaching meta data to a log entry, this data can be used to integrate with other services or capture context information.

Some examples include

A property to link log entries to Tickets

```
{
 "name":"ticket",
 "attributes":[
    {
       "name":"id",
       "value":"1234"
    },
    {
       "name":"URL",
       "value":"https://trac.nsls2.bnl.gov/ticket/1234"
    }]
}
```

1.5 events

There are instances when the log entry being created is actually associated with an event that happened some time ago. The users had higher priority tasks to address at that moment and is able to log the event after those tasks. The using **events** allows users to associate log entries with different instances in time, time based searches will ensure that these log entries are also found even if the create time might not fall in the search range.

CHAPTER

TWO

QUICK START

Download and install elasticsearch (verision 8.3) from elastic.com Download and install mongodb from mongodb

Configure the service The configuration files for Phoebus Olog are present under phoebus-olog/tree/master/src/main/resources/applications.properties

Build

```
mvn clean install
```

Start the service

```
mvn org.springframework.boot:spring-boot-maven-plugin:run
```

Detailed Installation Instructions: Install Phoebus Olog.

CHAPTER

THREE

REST API

3.1 Creating a Log Entry

Create a simple log entry

NOTE: deprecated, will be removed in future commits. Replaced by https://localhost:8181/Olog/logs/multipart **PUT** https://localhost:8181/Olog/logs

```
{
     "owner":"log",
     "description":"Beam Dump due to Major power dip Current Alarms Booster transmitter␣
↪switched back to lower state.",
     "level":"Info",
     "title":"Some title",
     "logbooks":[
        {
           "name":"Operations"
        }
     ]
}
```

Create a log entry, optionally with file attachments

PUT https://localhost:8181/Olog/logs/multipart

```
{
   "owner":"log",
     "description":"Beam Dump due to Major power dip Current Alarms Booster transmitter␣
↪switched back to lower state.",
     "level":"Info",
     "title":"Some title",
     "logbooks":[
        {
           "name":"Operations"
        }
     ],
     "attachments":[
        {"id": "82dd67fa-09df-11ee-be56-0242ac120002", "name":"MyScreenShot.png"},
        {"id": "c02948ad-4bbd-432f-aa4d-a687a54f8d40", "name":"MySpreadsheet.xlsx"}
     ]
}
```

NOTE Attachment ids must be unique, e.g. UUID. When creating a log entry - optionally with attachments - client **must**:

1. Use a multipart request and set the Content-Type to "multipart/form-data", even if no attachments are present.

#. If attachments are present: add one request part per attachment file, in the order they appear in the log entry. Each file must be added using "files" as the name for the part. #. Add the log entry as a request part with content type "application/json". The name of the part must be "logEntry".

Client must also be prepared to handle a HTTP 413 (payload too large) response in case the attached files exceed file and request size limits configured in the service.

Reply to a log entry. This uses the same end point as when creating a log entry, but client must send the unique id of the log entry to which the new one is a reply.

PUT https://localhost:8181/Olog/logs?inReplyTo=<id>

If <id> does not identify an existing log entry, a HTTP 400 status is returned.

Adding an attachment

POST https://localhost:8181/Olog/logs/attachments/{logId}

```
Content-Type: multipart/form-data; boundary=----formBoundary
------formBoundary
Content-Disposition: form-data; name="filename"
Content-Type: application/json
{"image1.png"}
------formBoundary
Content-Disposition: form-data; name="fileMetadataDescription"
Content-Type: application/json
{"image/png"}
------formBoundary
Content-Disposition: form-data; name="file "; filename="image1.png"
Content-Type: application/octet-steam
{...file content...}
------formBoundary--
```

3.2 Searching for Log Entries

GET https://localhost:8181/Olog/logs

Search Parameters

Keyword	Descriptions
Text search	
text	A list of keywords which are present in the log entry description
fuzzy	Allow fuzzy searches
phrase	Finds log entries with the exact same word/s
owner	Finds log entries with the given owner
Time based searches	
start	Search for log entries created after given time instant
end	Search for log entries created before the given time instant
includeevents	A flag to include log event times when
Meta Data searches	
tags	Search for log entries with at least one of the given tags
logbooks	Search for log entries with at least one of the given logbooks
Attachments searches	
attachments	To search for entries with at least one attachment
Pagination searches	
size	The number of log entries to be returned within each page
page	**The page number, i.e page 1 is the 1 to 1+size log** entries matching the search
Sorting Search Results	
sort	*up*|*down* order the search results based on create time

Example:

GET https://localhost:8181/Olog/logs/search?desc=dump&logbooks=Operations

The above search request will return all log entires with the term "dump" in their descriptions and which are part of the Operations logbook.

Retrieving an attachment of a log entry

GET https://localhost:8181/Olog/logs/attachments/{logId}/{filename}

Find entries with at least one attachment of type 'image'

GET https://localhost:8181/Olog/logs/search?attachments=image

3.3 Updating a Log Entry

POST https://localhost:8181/Olog/logs/{logId}

Update a log entry, the orginal log entry is archived in a seperate elastic index before any of the changes are applied.

Note: the create date, attachments, and events cannot be modified.

```
{
    "owner":"log",
    "description":"Beam Dump due to Major power dip Current Alarms Booster transmitter␣
```

(continues on next page)

(continued from previous page)

```
↪switched back to lower state.
                    New important info appended",
    "level":"Info",
    "title":"A new title",
    "logbooks":[
       {
          "name":"Operations"
       }
    ]
}
```

3.4 Managing Logbooks & Tags

Retrieve the list of existing tags

GET https://localhost:8181/Olog/tags

Retrieve the list of existing logbooks

GET https://localhost:8181/Olog/logbooks

Create a new tag

PUT https://localhost:8181/Olog/tags/{tagName}

```
https://localhost:8181/Olog/tags/Fault

{
     "name":"Fault",
     "state":"Active"
}
```

Create multiple tags

PUT https://localhost:8181/Olog/tags

```
https://localhost:8181/Olog/tags

[
  {"name":"Fault", "state":"Active" },
  {"name":"Alarm", "state":"Active" }
]
```

Create a new logbook

PUT https://localhost:8181/Olog/logbooks/{logbookName}

```
https://localhost:8181/Olog/logbooks/Operations

{
     "name":"Operations",
     "owner":"olog-logs",
     "state":"Active"
}
```

Create multiple logbooks

PUT https://localhost:8181/Olog/logbooks

```
https://localhost:8181/Olog/logbooks

[
  {"name":"Operations", "owner":"olog-logs", "state":"Active"},
  {"name":"DAMA",       "owner":"olog-logs", "state":"Active"}
]
```

3.5 Managing Properties

Retrieve the list of existing properties

GET https://localhost:8181/Olog/properties

Create a new property

PUT https://localhost:8181/Olog/properties/{propertyName}

```
{
     "name":"Ticket",
     "owner":"olog-logs",
     "state":"Active",
     "attributes":[
        {
           "name":"id",
           "state":"Active"
        },
        {
           "name":"url",
           "state":"Active"
        }
     ]
}
```

Create multiple properties

PUT https://localhost:8181/Olog/properties

```
[
  {
     "name":"Ticket",
     "owner":"olog-logs",
     "state":"Active",
     "attributes":[
        {"name":"id", "state":"Active"},
        {"name":"url", "state":"Active"}
     ]
  },
     {
     "name":"Scan",
     "owner":"olog-logs",
```

(continues on next page)

(continued from previous page)

```
    "state":"Active",
    "attributes":[
       {"name":"id", "state":"Active"}
    ]
  }
]
```

Javadocs

CHAPTER

FOUR

DEVELOPER DOCUMENTATION:

4.1 Change Log

A list of API changes, major features, and bug fixes included in each release.

4.1.1 service-olog-2.0.3 (current development version)

Date: TBD

- Add support for initializing default logbooks, properties, and tags on service startup
- Throw exception when invalid start and end time are requested in the log search parameters. Client will receive HTTP 400 (bad request) status.

4.1.2 service-olog-2.0.3

Date: Jan 10, 2022

4.1.3 service-olog-2.0.2

Date: Jan 9, 2022

- Remove vulnerabilities in dependencies

4.1.4 service-olog-2.0.1

Date: Dec 29, 2021

- Support for rich client text using markup

4.1.5 service-olog-2.0.0

Date: Oct 27, 2021

- First release of service on the phoebus framework - springboot framework
- Switch backend to elastic and mongodb