|  |  |  |
| --- | --- | --- |
|  | Facility for Rare Isotope Beams | msu.png |

ELog User Manual – Web Application

|  |  |  |  |
| --- | --- | --- | --- |
|  | Name | Signature | Date |
| Prepared By | Robert Gaul III |  |  |
| Reviewed By |  |  |  |
| Approved By |  |  |  |

Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| Version | Date | Author | Description |
| 1.0 | 9/1/2011 | Gaul | Initial |
| 1.1 | 1/16/2012 | Gaul | Update pictures, functionality descriptions |
|  |  |  |  |
|  |  |  |  |

Table of Contents

[1 Introduction 5](#_Toc314468792)

[2 Getting Started 5](#_Toc314468793)

[3 Creating a New Log 6](#_Toc314468794)

[4 Attaching a File 7](#_Toc314468795)

[5 Pasting a Screen Capture 8](#_Toc314468796)

[6 Adding IRMIS Components 9](#_Toc314468797)

[7 Editing an Existing Entry 11](#_Toc314468798)

[8 Sorting Existing Log Entries 13](#_Toc314468799)

[9 Searching for Log Entries 15](#_Toc314468800)

[10 Support 16](#_Toc314468801)

[11 Glossary 16](#_Toc314468802)

List of Figures and Tables

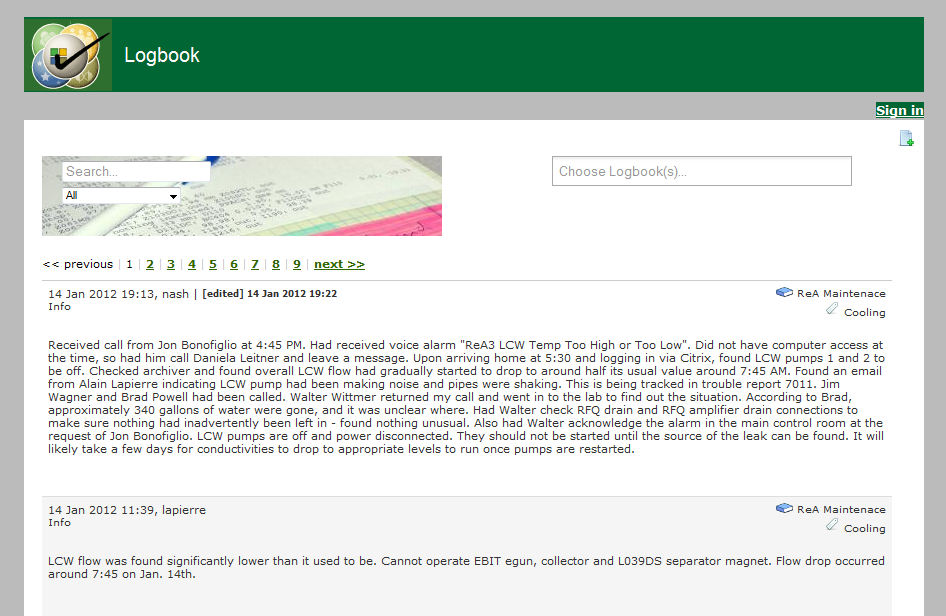
# Introduction

Many departments and groups within the Facility for Rare Isotope Beams (FRIB) and the National Superconducting Cyclotron Laboratory (NSCL) maintain a log of their operational and experimental findings. A Logbook is crucial to recording information acquired over time. This information usually consist of data, activities, and events; with the main purpose of having a history of what was known and when.

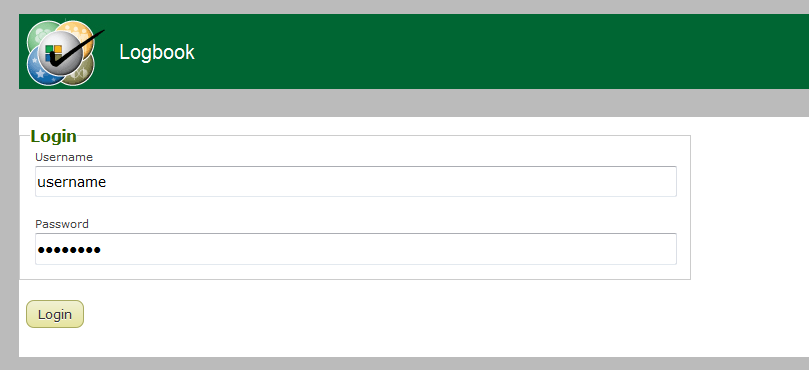
This manual details how to use the new web application.

# Getting Started

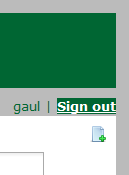
Elog is currently located at the following URL: <https://intra.nscl.msu.edu/cakeapps/logbook/olog/logs>



In order to use the functionality of Elog you must be signed in. Click on the “Sign in” link near the top right of the page and you will then be taken to a typical log-in screen where you will input your NSCL/FRIB username and UNIX password.



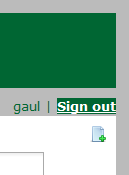
Upon successful authentication you will see your username next to a “Sign out” link near the top right of the page.



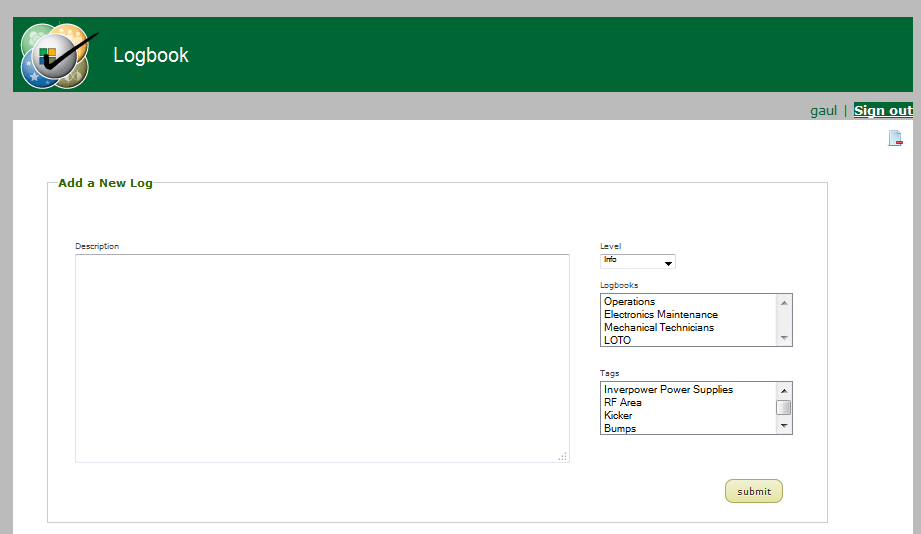
Please click the “Sign out” link when you are done using the application.

# Creating a New Log

Click on the following icon to create a new log:



This will expand a form that will be used to create your new log.



Here is an explanation of the editable fields:

* “Description” is the log entry
* “Level” is the type of log entry (is it general information, a problem, a request, etc?)
* “Logbooks” is a list of the available logbooks that your new log entry may go in. An entry may go into multiple logbooks, so the user may click on multiple logbook selections in this list.
* “Tags” is a list of identifiers that a log may be attributed with. A user may click on multiple selections here as well.

When you are done filling in the appropriate information click on the “submit” button. You will then see your new log entry on the index page.

# Attaching a File

You may attach a file (PDF, word, image, etc) to a logbook entry by clicking on the following button:



The standard operating system file browser will pop-up and you will be able to find and select the desired file. After completion you will see a link to the file below the log entry. Please note that this link will be a thumbnail for image uploads only.

# Pasting a Screen Capture

If you have a screen capture that you wish to attach to a log entry and do not want to have to save the image first then you may paste it into the web browser using a built-in Java applet. In order to do so click on the following button:

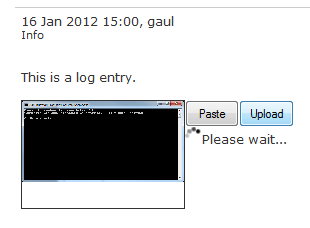


Please note: The applet might ask you to trust it, please ensure that you click the appropriate responses to do so.

Once the applet is loaded it has the following format:



Clicking on the “Paste” button will take the item currently in your systems clipboard (the screen capture) and paste it into the white box. Once you verify that the image in the box is correct you may then click “Upload” which will upload the image and attach it to the log entry.



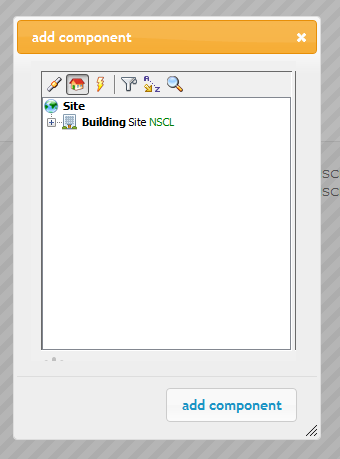
# Adding IRMIS Components

To add a component from IRMIS click on the following button:

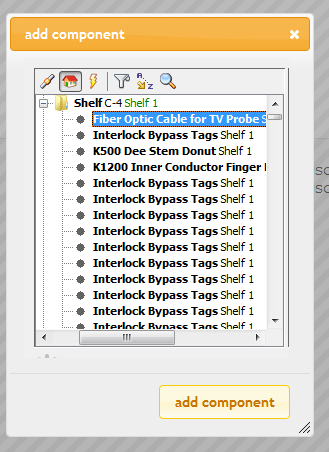


Please note: The applet might ask you to trust it, please ensure that you click the appropriate responses to do so.

Upon successful loading you will be shown the following applet:



Whatever you see in this applet is what is currently in IRMIS. You CANNOT add any component to IRMIS via this applet! What you can do is browse the various hierarchies (housing, network, power) or search for a component. Use the buttons at the top of the applet to find the component you wish to attach to the current log, click on it, and then click “add component”.



The page will refresh and you will see the component has been added to the log.

irmishierachy.png

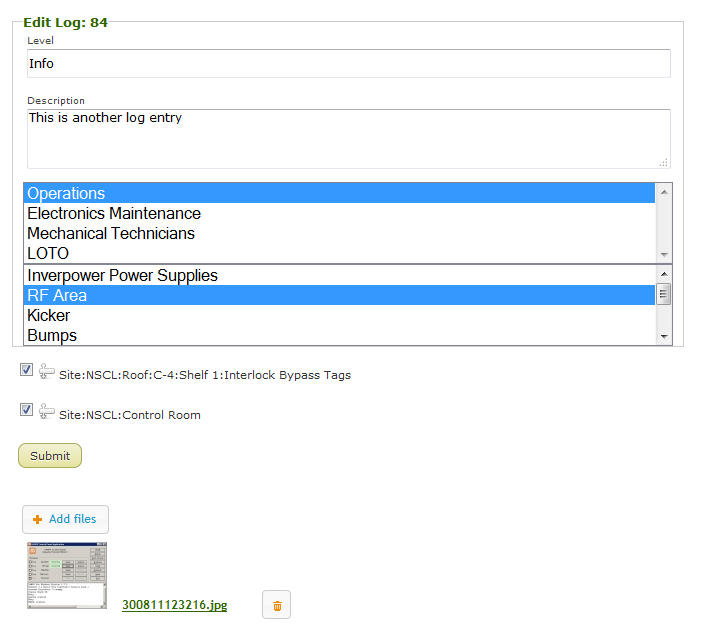
# Editing an Existing Entry

In order to edit an existing logbook entry click on the following button:



You will be brought to a form where you can edit all of the following metadata:

* Change the Level
* Change the Description
* Change the Logbook(s)
* Change the Tag(s)
* Add new file attachments using the file browser
* Delete existing file attachments
* Remove existing IRMIS components

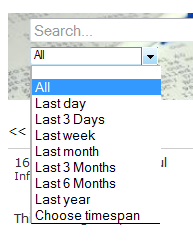


Click “submit” to save your changes.

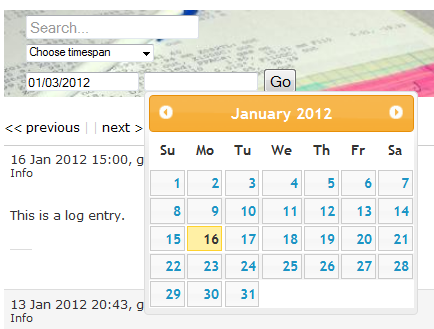
# Sorting Existing Log Entries

There are two ways to sort log entries: by time span or by logbook.

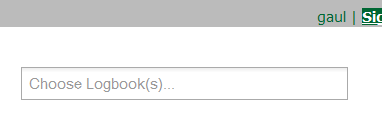
To sort by a time span click on the following dropdown box and click on the time span you wish to sort by.

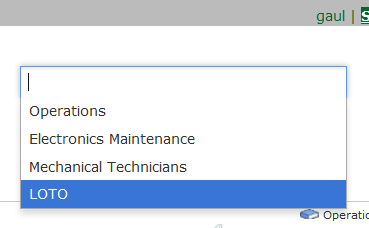


Choosing the “Choose timespan” option will allow the user to input a start date and an end date to search for log entries within those dates.

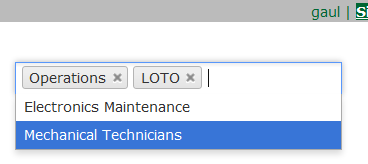


To view log entries from a given logbook click on the following dropdown and select the logbook you wish to view.





You may also filter by multiple logbooks.



# Searching for Log Entries

To search for a log entries given a keyword use the search bar and type a keyword you are looking for and press “Enter” on your keyboard to view the results.



# Support

Contact the following if you need help using Elog:

|  |  |
| --- | --- |
| Name | Email |
| Eric Berryman | berryman@nscl.msu.edu |
| Robert Gaul | gaul@nscl.msu.edu |

# Glossary

|  |  |
| --- | --- |
| Item | Description |
| Elog | Electronic Logbook Application |
| FRIB | Facility for Rare Isotope Beams |
| NSCL | National Superconducting Cyclotron Laboratory |

|  |  |  |
| --- | --- | --- |
|  | Facility for Rare Isotope Beams | msu.png |

ELog User Manual – CSS Plugin

|  |  |  |  |
| --- | --- | --- | --- |
|  | Name | Signature | Date |
| Prepared By | Robert Gaul III |  |  |
| Reviewed By |  |  |  |
| Approved By |  |  |  |

Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| Version | Date | Author | Description |
| 1.0 | 9/20/2011 | Gaul | Initial |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

Table of Contents

[1 Introduction 5](#_Toc304272858)

[2 Getting Started 5](#_Toc304272859)

[3 Creating a New Log 6](#_Toc304272860)

[4 Attaching a File 8](#_Toc304272861)

[5 Support 9](#_Toc304272862)

[6 Glossary 9](#_Toc304272863)

List of Figures and Tables

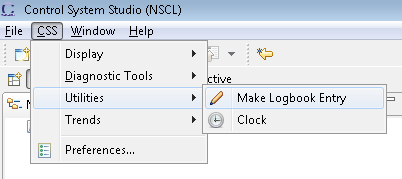
# Introduction

Many departments and groups within the Facility for Rare Isotope Beams (FRIB) and the National Superconducting Cyclotron Laboratory (NSCL) maintain a log of their operational and experimental findings. A Logbook is crucial to recording information acquired over time. This information usually consist of data, activities, and events; with the main purpose of having a history of what was known and when.

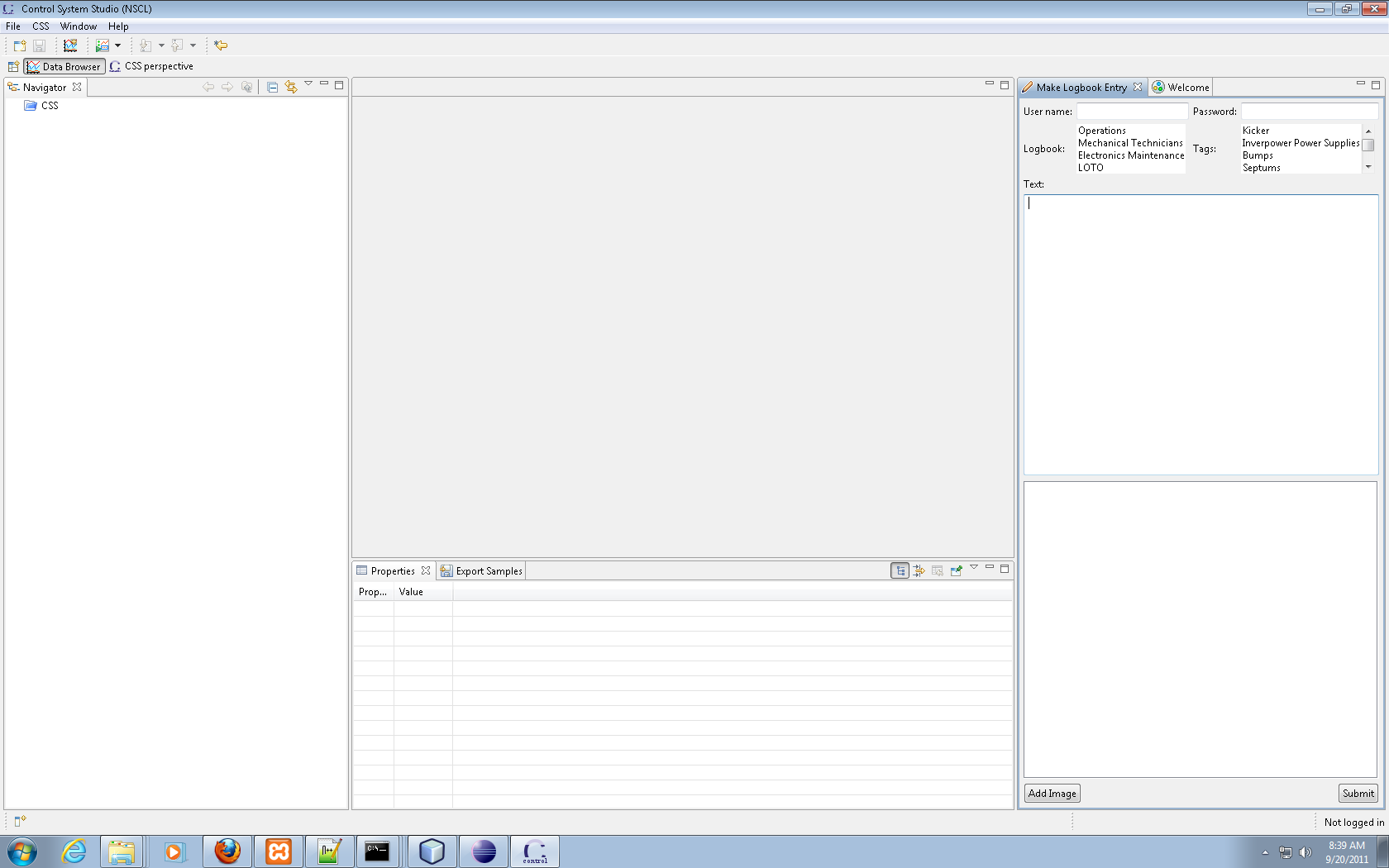
This manual details how to use the new Control System Studio (CSS) plugin.

# Getting Started

Start an instance of CSS. Click on CSS -> Utilities -> Make Logbook Entry



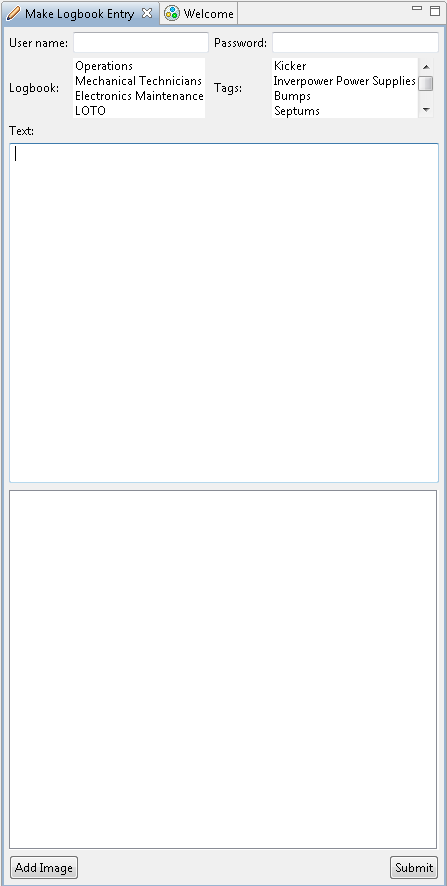
This will open up the Logbook plugin.



# Creating a New Log

Here is an explanation of the input fields:

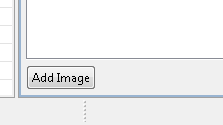
* “User name” is your NSCL/FRIB username
* “Password” is your UNIX password
* “Logbook” is a list of the available logbooks that your new log entry may go in. An entry may go into multiple logbooks, so the user may “CTRL + click” on multiple logbook selections in this list.
* “Tags” is a list of identifiers that a log may be attributed with. A user may click on multiple selections here as well.
* “Text” is the log entry
* The “Add Image” button allows you to upload and attach an image to the log entry



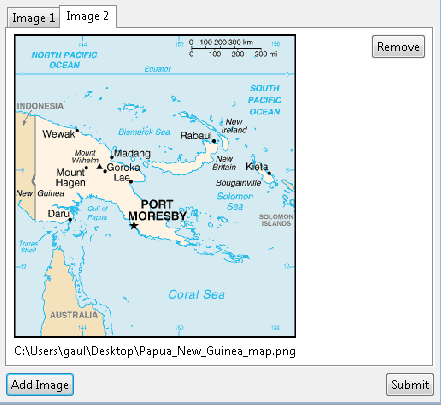
When you are done filling in the appropriate information click on the “Submit” button. You can then go to the Logbook web application and see your new log entry on the index page.

# Attaching a File

You may upload/attach a file (PDF, word, image, etc) to a logbook entry by clicking on the following button:



The standard operating system file browser will pop-up and you will be able to find and select the desired file. After completion you will see a preview of the file selected in the area provided. If you wish to remove the image you may do so by clicking on the “Remove” button. If you wish to add more images you may do so by clicking on “Add Image” button again and repeating the process. You will see multiple tabs for multiple images.



# Support

Contact the following if you need help using Elog:

|  |  |
| --- | --- |
| Name | Email |
| Eric Berryman | berryman@nscl.msu.edu |
| Robert Gaul | gaul@nscl.msu.edu |

# Glossary

|  |  |
| --- | --- |
| Item | Description |
| Elog | Electronic Logbook Application |
| CSS | Control System Studio |
| FRIB | Facility for Rare Isotope Beams |
| NSCL | National Superconducting Cyclotron Laboratory |

|  |  |  |
| --- | --- | --- |
| nscl.PNG | National Superconducting Cyclotron Laboratory | MSU-Wordmark-PMS-567.eps |

ELOG Software Architecture Document

|  |  |
| --- | --- |
|  |  |
| Project Name | Olog |
| Project Code | ELOG |
| Account |  |
| Department | Controls |
| Project Leader | Eric Berryman |
| Project Coordinator |  |

|  |  |  |  |
| --- | --- | --- | --- |
|  | Name | Signature | Date |
| Prepared By | Eric Berryman |  |  |
| Reviewed By |  |  |  |
| Approved By |  |  |  |

Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| Version | Date | Author | Description |
| 1.0 | 9/8/10 | Eric B | Initial |
| 1.1 | 9/28/11 | Rob G | Further refinement |
| 1.2 | 1/16/2012 | Rob G | Update sql diagram, add property information |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

Table of Contents

[1 Introduction 5](#_Toc304981076)

[1.1 Purpose 5](#_Toc304981077)

[1.2 Scope 5](#_Toc304981078)

[1.3 Definitions, Acronyms, and Abbreviations 5](#_Toc304981079)

[1.4 References 6](#_Toc304981080)

[1.5 Overview 6](#_Toc304981081)

[2 Architectural Design 6](#_Toc304981082)

[2.1 Design Overviews 6](#_Toc304981083)

[2.1.1 Component Overview 6](#_Toc304981084)

[2.1.2 SQL Overview 7](#_Toc304981085)

[2.1.3 Class Overview 8](#_Toc304981086)

[2.1.4 Class Detail 9](#_Toc304981087)

[2.1.5 Deployment Overview 17](#_Toc304981088)

[3 Authentication, Authorization 17](#_Toc304981089)

[4 Implementation Strategy 17](#_Toc304981090)

# Introduction

National Superconducting Cyclotron Laboratory (NSCL) is a world leader in rare isotope research and nuclear science education. NSCL scientists and researchers conduct advanced research in nuclear science, nuclear astrophysics, and accelerator physics. To facilitate this research, NSCL operates multiple particle accelerators. A Logbook is a critical element used in order to record information acquired over time. This information usually consists of data sets, ongoing activities, and events occurring; with the main purpose of having a history of what was known at a certain time.

The current logbook application (Hourlog) has grown out of scope from its original requirements and has become difficult to maintain because of the ad-hoc programming methods used at its creation and being dependent on using direct database calls. This project will create a new logbook application, with a flexible plugin-based framework, that can meet future requirements through the use of additional plugins.

## Purpose

The purpose of this document is to describe the high-level design for ELOG’s implementation. It is intended for the following audience:

* Members of the Controls and Computing department that are overseeing the project; they need to make sure that the design fulfills the requirements
* Members of the design team at CC;
* Members of the software development team at CC; they need to make sure that the design is accurate and feasible

## Scope

Refer to Section 2 of the ELOG Project Plan [6] for the scope of the system.

## Definitions, Acronyms, and Abbreviations

Table Definition, Acronyms, and Abbreviations

|  |  |
| --- | --- |
| Item | Description |
| CC | FRIB’s Controls and Computing Department |
| EPICS | Experimental Physics and Industrial Control System |
| NSCL | National Superconducting Cyclotron Laboratory |
| PL | Project Leader |
| PC | Project Coordinator |
| POC | Proof of Concept. A Prototype. |
| PV | Process Variable. Identifier for a data item in EPICS. |
| Quark | EE’s Process Infrastructure Web Portal https://intra/departments/electronics/quark |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

## References

1. Quark, The NSCL EE Process Infrastructure, <https://intra.nscl.msu.edu/departments/electronics/quark/quark.php>

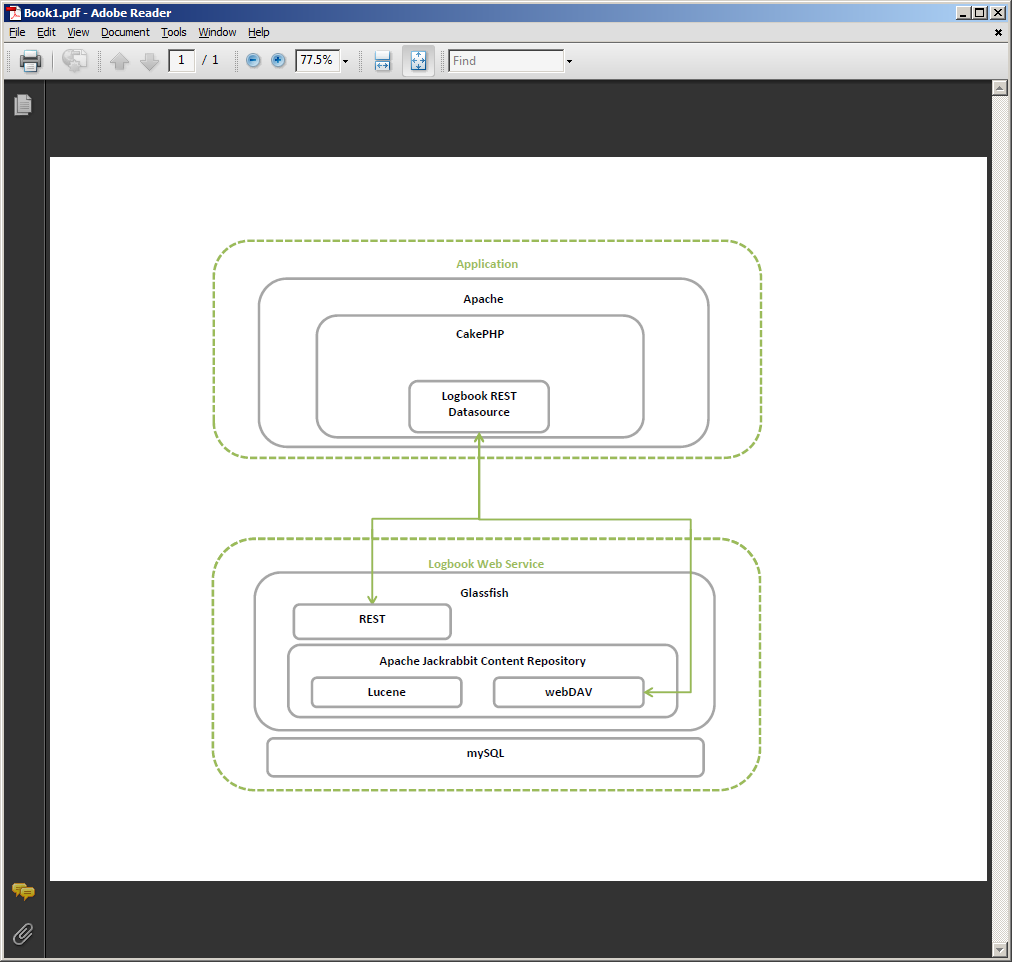
## Overview

The next section, *Architecture*, describes the high-level design for the system.

# Architectural Design

## Design Overviews

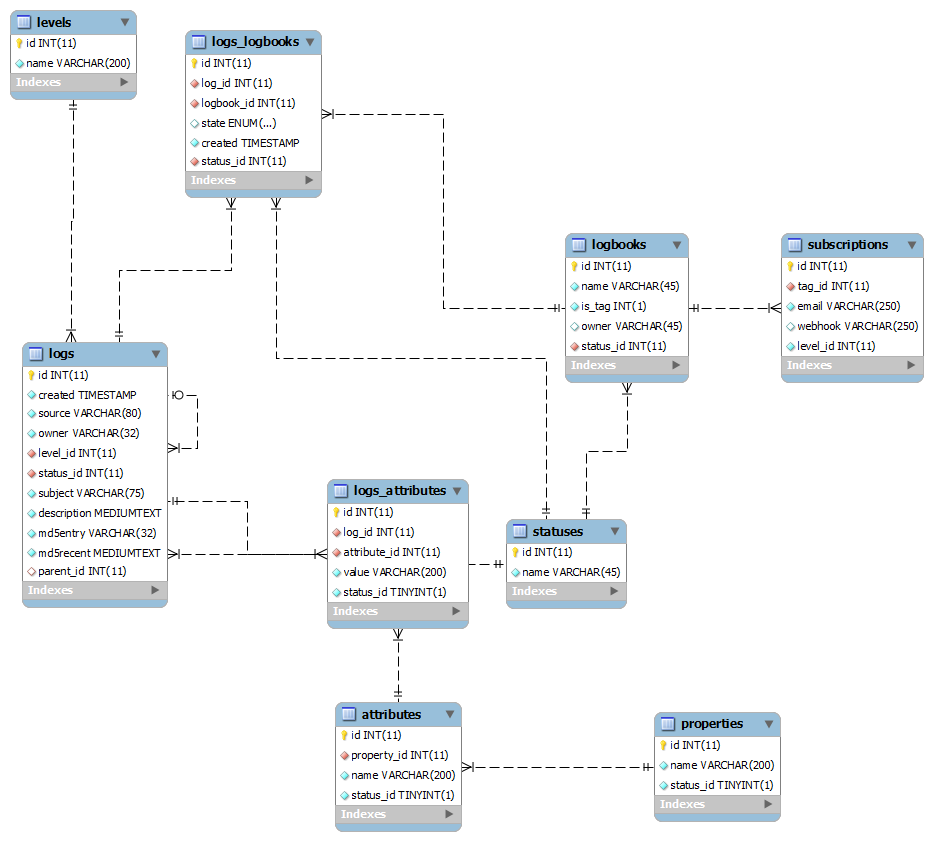
### Component Overview



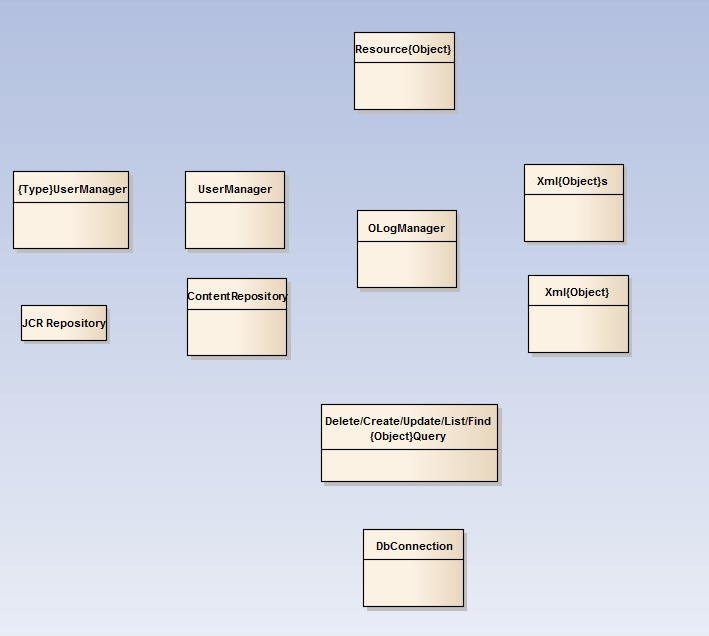
The technologies used for ELOG include the following:

* A web application
  + Built with CakePHP
  + Uses a REST datasource
* A web service
  + Built on Glassfish
  + Uses Apache Jackrabbit repository to store attachments
    - Lucene is used for searching
  + All log data goes into a MySQL database

### SQL Overview



### Class Overview



A Resource is a class that contains the operations for handling data relevant to a logbook (tags and log entries for example) in a RESTful way. These resources are connected to a specific URI/URL and HTTP request type.

A UserManager is a class that will manage how groups a user is in are found. Various methods can be used so there will be a super-class that may be inherited from.

In order to facilitate maintainability and easier communication of data between the client and the server XML is parsed into XML classes containing data contained in the XML document.

Files that are attached to a particular log will be held in a “ContentRepository”. An example is the Jackrabbit content repository. Because there are many possible repositories there will be a “ContentRepository” super-class that will be inheritable and thus the use of any repository that meets the needs of keeping attachments can be used.

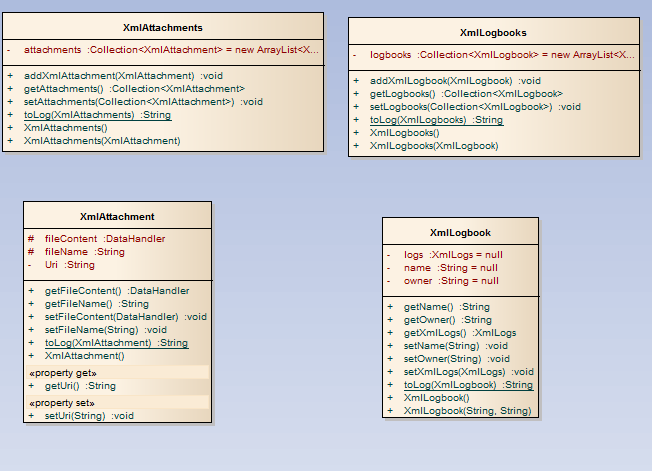
All CRUD (Create, Read, Update, and Delete) functions related to data will be defined once in specific classes representing the operation performed (such as finding logs or deleting logbooks). These definitions will be MySQL database queries connected to a database defined in a JNDI and maintained by “DbConnection”.

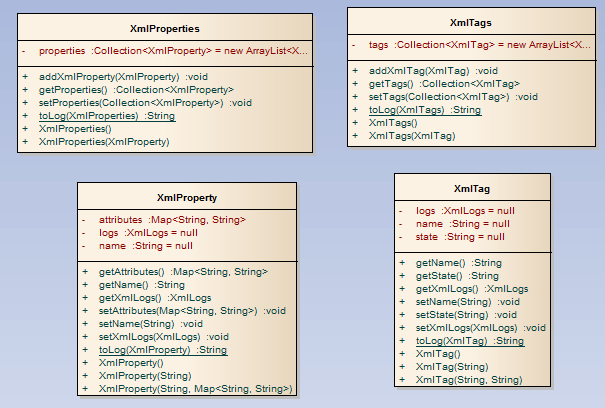
In order to keep all of these classes coherent an “OLogManager” is used as the central calling point in order to keep a reasonable flow when interacting with the other classes.

### Class Detail



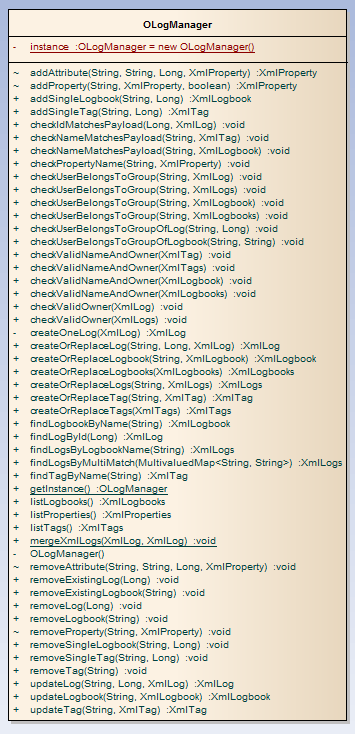
Each resource has CRUD operations that map to a given URI. HTTP requests are sent to these URI’s in order to add, remove, update, or delete data. All of the logic, after these requests are sent, is contained within these classes. All resources are connected to a logger that will be useful for debugging purposes.



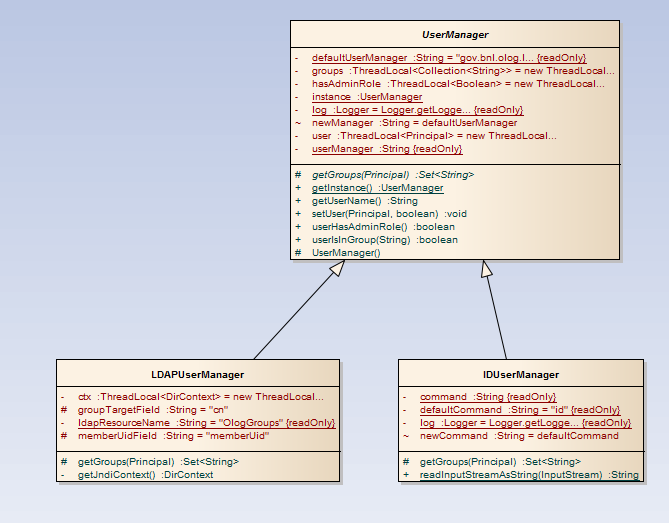




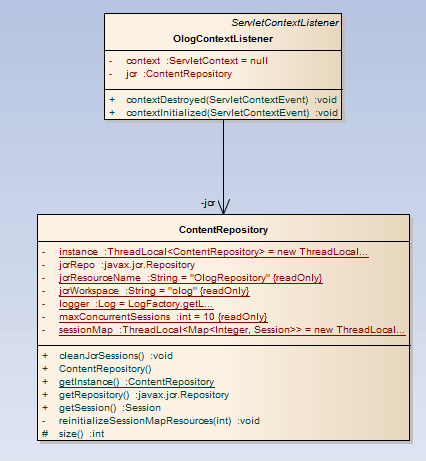
When sending HTTP requests (such as a PUT) the user is required to pass XML to the server. Similarly every time information is received (through a GET request) this information is received as XML. In both cases this XML needs to be parsed into a Java object that allows the information to be easily retrieved by the server and easily parsed through by the client. The server and client will use XML classes to deal with data being passed around.

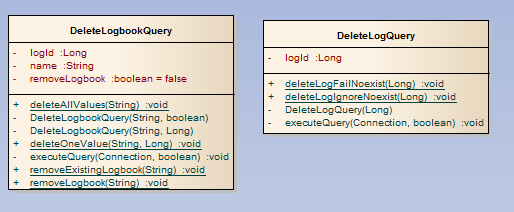


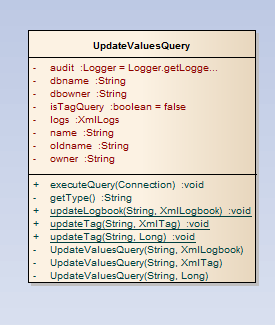
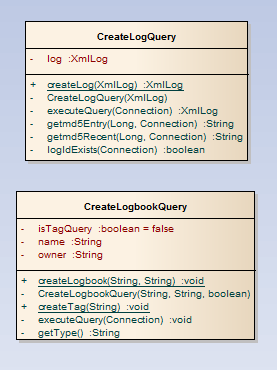
All calls essential to the ELOG service will be handled through a manager layer. This includes all calls to the Querying classes and calls to validate a user.

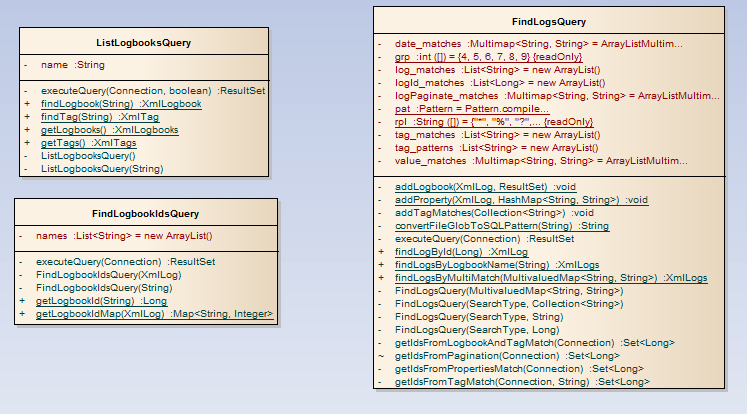


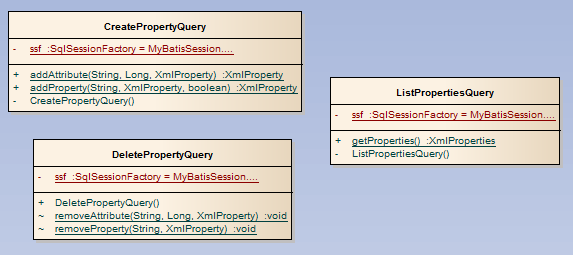
The “UserManager” super-class contains all of the attributes and all of the methods that are to be inherited by children classes when managing user groups. Each implementation differs in its way that groups per user are retrieved. There is an implementation that binds to an LDAP JNDI resource (“LDAPUserManager”) and an implementation that uses the UNIX “id” command (“IDUserManager”).



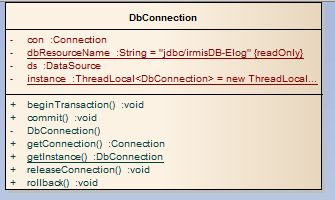




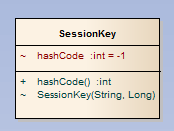


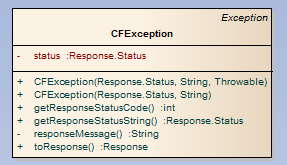


All queries that get, delete, update, or edit data in the underlying database are contained within these classes. This allows for higher maintainability due to queries being set at one location.

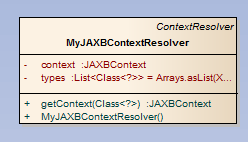


This class maintains the connections to the database containing all ELOG information.



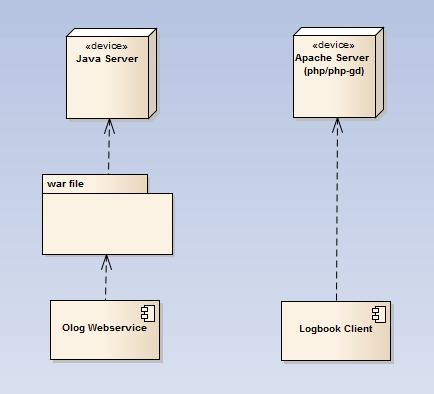


An exception class made specifically for ELOG exceptions.



This class will handle the marshaling of JSON into a Java object.

### Deployment Overview



To deploy the Olog webservice a WAR file needs to be built and deployed on a Java-based server (such as Glassfish or JBOSS). To deploy the Logbook CakePHP client the code needs to be deployed to a production environment and Apache needs to be configured to point to the new code.

#### Configuration Files

bootstrap.properties: set repository home directory

repository.xml: set data storage type for repository (default: Derby)

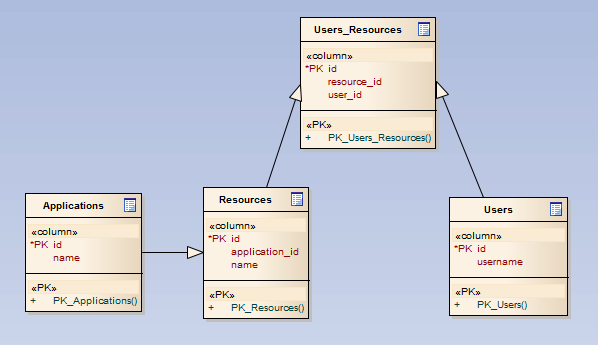
sun-web.xml: set role mapping

olog\logbook\app\config\database.php: set webservice URIs for client

# Authentication, Authorization

Authentication is handled by JAAS. JAAS can be attached to the system administrator’s preferred authentication service in the Java server configuration. The NSCL Olog JAAS will point to AD LDAP.

Authorization, as mentioned in section 2.1.4, is handled by the UserManger super-class. There is an implementation that binds to an LDAP JNDI resource (“LDAPUserManager”) and an implementation that uses the UNIX “id” command (“IDUserManager”). The NSCL Olog will implement and use a new class (“DatabaseUserManager”). The database will have four tables; Users, Resources, Applications, Users\_Resources. The table Users will consist of usernames. The table Applications will consist of application names (ie. in this case olog). The Resources table will name the controlled resources (ie. logbook/Operations). The Users\_Resources table will map the users to the controlled resource; if the user-resource map exists, then the user can access the controlled resource. The All tables will need to be filled in by hand; an application front end can be made to simplify data entry.



# Implementation Strategy

1. The Olog service will be a RESTful (stateless) web service. Data will be in the form of XML or JSON, depending on HTTP headers in request (“content-type”, “accepts”). All business logic will be contained within the web service.
   1. API
      1. HTTP Methods
         1. GET: retrieve or find
         2. PUT: create or update, replacing element
         3. POST: create or update, subordinates element
         4. DELETE: delete element
      2. Ownership Restrictions
         1. Logs require a user owner, while Logbooks require a group owner.
         2. Only users that belong to the Logbook group can make or edit entries in that Logbook.
         3. There are four groups that are specified in the sun-web.xml file for group association; admin, logbook, tags, logs. If a user is in a particular group, they are allowed to add/modify/delete that resource (admin being all).
      3. POST/PUT payload (PUT is a single <log/>)

**<?xml version="1.0" encoding="UTF-8" standalone="yes"?>**

**<logs>**

**<log createdDate="2011-09-09T02:18:04-04:00" id="568" level="Info" owner="me" version="0">**

**<description>This is an entry about things.</description>**

**<subject>Logbook Application</subject>**

**<attachments/>**

**<logbooks>**

**<logbook name="ReA Maintenace" owner="reamntg"/>**

**<logbook name="ReA Operations" owner="r3logbkg"/>**

**</logbooks>**

**<properties>  
   <property name="component">  
       <attributes>  
           <entry>  
               <key>type</key>  
               <value>coupler</value>  
           </entry>  
           <entry>  
               <key>fieldname</key>  
               <value>coupler 1</value>  
           </entry>  
       </attributes>  
   </property>  
   <property name="component">  
       <attributes>  
           <entry>  
               <key>type</key>  
               <value>coupler</value>  
           </entry>  
           <entry>  
               <key>fieldname</key>  
               <value>coupler 2</value>  
           </entry>  
       </attributes>  
   </property>  
</properties>**

**<tags>**

**<tag name="Cryomodule/cryostat"/>**

**</tags>**

**</log>**

**</logs>**

* + 1. URLs
       1. …/logs?logbook=Operations&tag=cryostat&componentType=Room
          1. GET: query with parameters
          2. Wildcard characters “\*” and “?” are allowed
          3. Logbooks are “OR-ed”, while other parameters are “AND-ed”
       2. …/logs
          1. POST: Create or update entries with payload (<logs/>); returns entries with ids.
       3. …/logs/{id}
          1. GET: Returns entry with id
          2. PUT: Create or replace entry with payload (<log/>)
          3. DELETE: Deletes the entry with the given id (marked delete)
       4. …/logbooks
          1. GET: List Logbooks
          2. POST: Create Logbooks (<logbooks/>
       5. …/logbooks/{name}
          1. GET: Returns matching single Logbook
          2. PUT: Create or replace Logbook with the given name
          3. DELETE: Deletes the Logbook (marked delete)
       6. …/tags
          1. GET: List Tags
          2. POST: Create Tags (<tags/>
       7. …/tags/{name}
          1. GET: Returns matching single Tag
          2. PUT: Create or replace Tag with the given name
          3. DELETE: Deletes the Tag (marked delete)
       8. …/properties
          1. GET => get all property Names
       9. …/properties/{propName}
          1. PUT => Associate attribute(s) with property (Destructive)
          2. POST => Same, not destructive
          3. DELETE => Remove attribute from property (set as inactive)
          4. GET => View attributes associated with a property
       10. …/properties/{propName}/{logId}
           1. PUT - add the payload property
           2. DELETE - remove the property from the log
  1. Repository
     1. Apache Jackrabbit was chosen for the repository to store attachments, embedded in the Olog service. By default Derby is the data storage for Apache Jackrabbit, but MySQL can also be used.
     2. A WEBDAV service exposed through Jackrabbit will be used as an interface to upload/download attachments. File structure will be; directories are named based upon the log entry id, thumbnails are in the directory “thumbnail”, and then sub-directories based upon log entry id.
     3. Thumbnails will be supplied by the client and not the user.
  2. Database
     1. Logs have a parent-child relationship in which the parent is the original entry and children are edits of that entry. Xml from the service reflect this by having all logs keep the parent id as their entry id; as an edit, children have a version number which increments.
     2. Logbooks/Tags are attached to Logs with a “Toxi” relationship (using a link table to represent the relationship between Logs and Logbooks/Tags). Tags and Logbooks are the same from a database perspective except that Logbooks require an owner whereas Tags don’t. Tags are marked with the is\_tag flag differentiate themselves from Logbooks.
     3. Properties are a one-to-many relationship, allowing multiple key/value pairs to be associated with an entry. If this relationship slows queries too much, key/value pairs will be moved to the Jackrabbit repository as attributes to the log entry id node.
     4. On the creation of an entry, the entry is hashed (with the md5recent column) and placed in the md5entry column, and then the past 10 hash entries are placed in the md5recent column. This is to verify that entries and their histories have not been modified.
     5. Subscriptions will allow a person to enter an email address or “web hook” in order to be notified when a specific Logbook or Tag receives an entry with a Level higher than or equal to what is stated in the subscription.

1. The Logbook client will be a web application using a plugin-based framework which will produce a more maintainable logbook with clean interfaces to dependencies that can be flexible to future needs.
   1. CakePHP was chosen as the PHP framework for the reason of proficiency in both the Computers Department and the CC departments.
   2. All added code should be done through plugins. The logbook base application is the plugin “olog”. The “PluginManager” was added to improve upon CakePHP’s plugin framework. The “PluginManager” allows each plugin to have its own API and hooks to communicate with other plugins.
   3. The logbook client holds a user session open on login using the CakePHP “auth” framework.