`TESDAQ` Statement of Work and Design Specification

Connor Duncan, Aaron Elersich

November 27, 2019

Contents

1 Project Objectives

The objective of this project is to provide a self-contained Data Acquisition and Control System for Dark Matter Search using Phonon Detectors. `TESDAQ` should provide a method for storing and tracking large binary files from acquisition, as well as desired metadata for these files. It also should provide a method for tracking and altering the state of various acquisition devices using standard web technologies. Real-time streaming of large volumes of data should be possible via a specified interface.

2 Requirements

1. Database which tracks run metadata and file location.
2. Standard, documented file structure for storage of data.
3. RESTful API for reading run database, downloading either chunks of or individual files.
4. In-memory store of current device state
5. Standard, language-agnostic object for representing device.
6. RESTful API for CRUD current state.
7. Socket availability for real-time streaming of data.
8. Authentication methods and user database with multiple levels of access.

3 Software Design Description

The project is broken into two largely independent parts, the Database, and the State. The database will manage archival work and integrations into offline analysis, as well as storage of authentication tokens and permission levels. The State will manage the storage, access and alteration of ongoing acquisition activities, as well as the streaming of data in real time to multiple clients.

3.1 Database

3.1.1 File Organization

Store data as `<Start>-<Finish>.hd5` where `<Start>`, `<Finish>` are given by strings representing `YYMMDDHHmmSS`. Each HDF5 file represents a unique run. Each file will contain the following structure, following the Scientific Data Exchange Format.[1]

[1]doi:10.1107/S160057751401604X

HDF5 Object	HDF5 Object Type	Description
exchange	Group	Contains measured data tables and metadata table.
implements	Group	Contains list with implemented tables.
⋮	Group/Tables	Implemented/derived tables.

3.1.2 Metadata Tracking

3.1.3 REST API

`/database/runs` Returns JSON-formatted list of run metadata. Can be sorted by fields using standard GET search parameters.

`/database/runs/uuid` Returns JSON-formatted list of file information for run `uuid`, including size, chunk size, implemented tables.

3.2 State

3.2.1 State Access

3.2.2 State Control