CS CAPSTONE REQUIREMENTS DOCUMENT

NOVEMBER 3, 2017

COVERAGEJSON RESPONSE HANDLER FOR OPENDAP

PREPARED FOR

NASA JET PROPULSION LABORATORY

LEWIS JOHN MCGIBBNEY

PREPARED BY

GROUP55

RILEY RIMER
RIVER HENDRIKSEN
COREY HEMPHILL

Abstract

This document defines the requirements necessary to develop the CoverageJSON Response Handler for OPenDAP.

1 Introduction

1.1 Purpose

The purpose of this project is to enhance the usability of OPeNDAP by integrating it with CoverageJSON. This new response handler implementation will expand functionality for all users of both OPeNDAP and CoverageJSON, and will be particularly useful to NASA Jet Propulsion Laboratory.

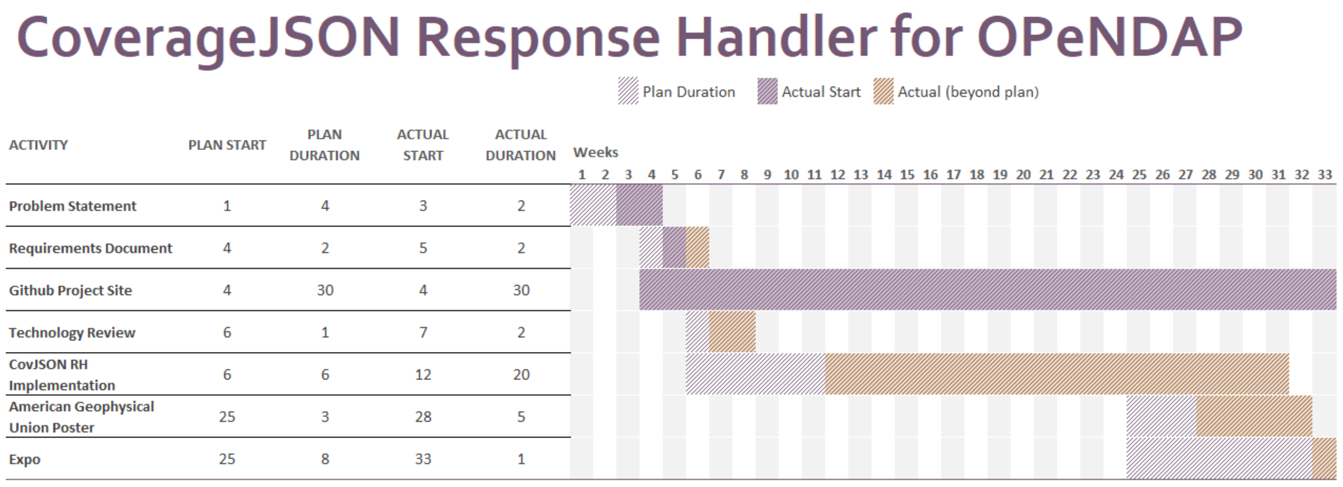
1.2 Scope

The result of this project will be a CoverageJSON response handler for OPeNDAP. This will allow for OPeNDAP to serve users data in the CoverageJSON data format, which will allow users to view their data as a coverage, rather than the scientific formats currently implemented in OPeNDAP.

1.3 Time Commitment Expectations

1) Individual Time Commitment
    a) During the implementation phase of the project, team members will commit a minimum of 6 hours a week to the project. This is includes programming, testing, documentation, and any other necessary task.

2) Gantt Chart



ACTIVITY	PLAN START	PLAN DURATION	ACTUAL START	ACTUAL DURATION
Problem Statement	1	4	3	2
Requirements Document	4	2	5	2
Github Project Site	4	30	4	30
Technology Review	6	1	7	2
CovJSON RH Implementation	6	6	12	20
American Geophysical Union Poster	25	3	28	5
Expo	25	8	33	1

1.4 Definitions

1) OPeNDAP - Open Source Project for a Network Data Access Protocol
2) CoverageJSON - JSON data format for encoding coverage data
3) JSON - JavaScript Object Notation
4) NASA JPL - The National Aeronautics and Space Administration Jet Propulsion Laboratory
5) Response Handler - the closure that is executed to parse the HTTP response that is returned from the server

1.5 References

1) OPeNDAP Advanced Software for Remote Data Retrieval https://www.opendap.org/
2) CoverageJSON https://covjson.org/

2 Overall Description

2.1 Product Perspective

The CoverageJSON response handler will be incorporated into the larger OPeNDAP open-source software project. Therefore, all function calls will be similar to those implemented in OPeNDAP. The current OPeNDAP JavaScript call looks like this:

```
createDataRequestForm("url" : "http://test.com/data.gz", "containerID" : "requestform");
```

The CoverageJSON handler will be similar in execution to remain faithful to the requirements set by OPeNDAP.

2.2 Product Functionality

The CoverageJSON response handler will have the same functionality as the response handlers already implemented in OPeNDAP.

1) Data that is converted to CovJSON will contain the same information as the source, but in the CovJSON format.
2) Users will be able to obtain data via a GUI that is already implemented in OPeNDAP, however, there will be a new option for CovJSON.
3) The CovJSON response handler will be integrated into OPeNDAP's source Github.

2.3 User Characteristics

The expected characteristics of a user of the CovJSON response handler will be the same as the characteristics of an expected OPeNDAP user. These characteristics include the following:

1) Scientists looking to share coverage data over the Internet.
2) Groups looking to provide compatible clients, servers, and SDKs.
3) Users looking to conform to the NASA community standard.

2.4 Assumptions and Dependencies

1) There will be an adequate amount of accurate documentation on OPeNDAP and its response handlers.
2) There will be a Hyrax server environment capable enough to test on.
3) There will be feedback on the testing and documentation needed to have the response handler pulled into the OPeNDAP project.

3 Requirements

The CoverageJSON response handler will need to fulfill the following requirements:

1) Handle and feed out data in the CoverageJSON format.
2) All handled data should match the original source data exactly.
3) Able to be pulled into the OPeNDAP source project.
4) The handler will need to be able to be called in the same fashion that the current OPeNDAP handlers are called, and should behave similarly.
5) The handler must be fully testable and should meet the standards defined by the client.
6) All code should adhere to strict coding standards defined by the client.