

CCSITM
Carbon Capture Simulation Initiative

Data Management Technology Survey and Recommendation

Prepared by:
Tom Epperly
LLNL

Prepared for
U.S. Department of Energy
National Energy Technology Laboratory

September 27, 2013



U.S. DEPARTMENT OF
ENERGY

Revision Log

| Revision | Date | Revised By: | Description |
|----------|-----------|-----------------------------|-------------------------------|
| V0.0 | 7/1/2013 | Tom Epperly | First edition |
| V0.1 | 8/1/2013 | Tom Epperly and Deb Agarwal | Minor Edits |
| V0.2 | 9/27/2013 | Tom Epperly | LLNL disclaimer & minor edits |
| | | | |

Disclaimer

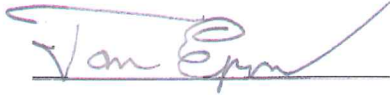
This report was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government nor any agency thereof, nor any of their employees, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government or any agency thereof. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States Government or any agency thereof.

This document was prepared as an account of work sponsored by an agency of the United States government. Neither the United States government nor Lawrence Livermore National Security, LLC, nor any of their employees makes any warranty, expressed or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States government or Lawrence Livermore National Security, LLC. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States government or Lawrence Livermore National Security, LLC, and shall not be used for advertising or product endorsement purposes.

This work performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under Contract DE-AC52-07NA27344. LLNL-TR-643636

Approvals

Author:



9/27/2013
Date

Tom Epperly
CCSI Data Management Lead

Approve:



9/30/2013
Date

Deb Agarwal
Task 5 Lead

Concur:



9/30/2013
Date

David Miller
CCSI Technical Team Lead

Table of Contents

| | | |
|-----|---|----|
| 1 | Introduction..... | 5 |
| 1.1 | CCSI Project Overview | 5 |
| 1.2 | Background..... | 5 |
| 1.3 | Data Management Requirements..... | 5 |
| 2 | Survey of Technologies | 6 |
| 2.1 | Velo..... | 6 |
| 2.2 | Alfresco Community Edition 4.2.c (December 2012)..... | 7 |
| 2.3 | Apache Jackrabbit 2.6.1..... | 8 |
| 2.4 | HDF5 1.8.11 | 9 |
| 2.5 | JSON..... | 10 |
| 2.6 | XML..... | 11 |
| 3 | Recommendations..... | 12 |

1 Introduction

The Data Management Plan (DMP) document already released describes the architecture and plan for developing the data management framework needed by the Carbon Capture Simulation Initiative (CCSI) toolset. The CCSI toolset includes: experimental data, physical property methods and parameters, empirical correlations, validation data set, models, simulation inputs and outputs, reduced-order models, cost models, risk models, uncertainty estimation, etc. Each tool in the toolset requires and generates data, and the data management system organizes and stores the data. The data management framework will be driven by end-users and interact with CCSI workflow components such as Turbine. This document reports on the technology survey completed as the first step in the Data Management Plan. It also provides recommendations for a path forward based on the survey results.

1.1 CCSI Project Overview

The *Carbon Capture Simulation Initiative* (CCSI) is a partnership among national laboratories, industry and academic institutions that will develop and deploy state-of-the-art computational modeling and simulation tools to accelerate the commercialization of carbon capture technologies from discovery to development, demonstration, and ultimately the widespread deployment to hundreds of power plants. By developing the *CCSI Toolset*, a comprehensive, integrated suite of validated science-based computational models, this initiative will provide simulation tools that will increase confidence in designs, thereby reducing the risk associated with incorporating multiple innovative technologies into new carbon capture solutions. The scientific underpinnings encoded into the suite of models will also ensure that learning will be maximized from successive technology generations.

The CCSI Elements are responsible for the design, implementation, verification and validation of simulation software codes that support Carbon Capture Simulations which allow new concepts to move from the lab to power plant quickly and at low cost.

1.2 Background

The Data Management Framework must interact with CCSI toolset and commercial tools and technologies. This section describes the tools and technologies deemed relevant, due to alignment with CCSI needs, adoption by current CCSI collaborators, or both.

1.3 Data Management Requirements

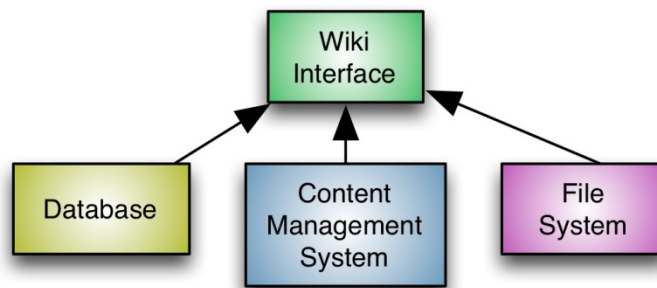
The CCSI data management system has the following requirements:

- Supports all existing types of CCSI related data and is expandable to handle other data as needed
- Stores provenance information when available
- Provides versioning of stored information and storage of past versions
- Allows user to store data as files when desired (outside the data management system)
- Able to extract data from the data management system in its original data format (i.e., one of the supported data model formats)
- Provides programmatic interfaces to allow tools to store and retrieve data directly
- Deployable as a system to allow industry partners to run their own data management system
- Able to configure multiple, independent data management system instances at the same industry
- Supports authentication and authorization of users and access control for information and files
- Provides a search interface to help find data in the data management system
- Easy to install and configure

In the situation where the data management system is being installed to interact with an industry partner's own data management system, some custom configuration and integration will be required. Our requirement is to provide well-documented interfaces and examples to show how the CCSI Data Management System can be connected to a third-party system. We will investigate and leverage open standards for authentication such as OpenID to enable the possibility for reusing an industry partner's preexisting authentication systems. Such capabilities will require additional customization.

2 Survey of Technologies

The data management system is designed to provide the persistent storage of simulation versions, results, decisions made, and configurations tried. The CCSI data model that has already been published, defines the types and formats of the data in the CCSI system. The data management system will be composed of several parts (see Figure below).



CCSI Data Management System Components

The following sections provide survey results for each of the technologies considered for inclusion in the CCSI data management framework.

2.1 Velo

Velo is based on Alfresco 3.4.c (released in December 2010) with some additional features written at the Pacific Northwest National Laboratory (PNNL). Currently, an installation package is available for Velo as the Reveal Server installation from the CCSI software downloads page.

2.1.1 Installation

The installation of Velo from start to a working server took about 2 weeks of calendar time. The installation wasn't a full time effort, and I was working on other projects at the same time. I required help from PNNL on 4 issues, and each issue was resolved with an exchange of emails. Most issues were resolved within 24 hours of reporting them.

2.1.2 Capabilities

After getting Velo installed, I was able to use the web interface to store and download files. However, most of the functionality that's apparent is based on Alfresco. I requested information about how to use the extra features provided by Velo – for example developer documentation or help on how to access the Velo-specific features of the system, and the PNNL staff were unable to help me get access to that kind of material, at least during the period of time I had available to evaluate Velo. At this point, it seems that

they have the infrastructure to support Velo as part of the Reveal software package but not as a tool for developing the CCSI data management framework.

Alfresco's capabilities will be described below in the evaluation of Alfresco.

2.1.3 Documentation & Web Presence

There is no significant web presence for Velo. Searching for “Velo” on the www.pnl.gov website produces 24 hits (based on web content May 2013). Many of these hits are redundant references to a Velo paper at ACM. Velo has a source code repository and bug tracking system, but neither is publicly accessible. There is no public project page or significant documentation available currently.

2.1.4 License

Velo is available to CCSI participants under the CCSI T&E license. The licensing of the Alfresco component of Velo is covered below.

2.1.5 Assessment

Based on presentations, Velo has significant capabilities. However, these capabilities do not appear to have sufficient developer documentation to be useful outside of PNNL. The capabilities that are readily apparent are those of Alfresco. Despite being a tool that supports several PNNL projects, the tool has no web presence, documentation, or bug tracking system. In summary, it does not seem mature enough at this point to serve as a key component of the CCSI document management framework.

2.2 Alfresco Community Edition 4.2.c (December 2012)

This is the latest edition of the Alfresco

2.2.1 Documentation & Web Presence

Alfresco is supported by alfresco.com. alfresco.com has information about a commercially supported version of Alfresco that can be hosted on your machines or on cloud resources (presumably internet accessible systems managed and supported by Alfresco). alfresco.com has training, end-user documentation, developer documentation, developer blogs, discussion forums, and software downloads.

2.2.2 Installation

The Alfresco community release is available for download from www.alfresco.com/products/community

The 469MB installation comes with Java, PostgreSQL, SharePoint protocol support, Google Docs integration, and LibreOffice. I configured the system to use MySQL (a non-standard option), and the whole operation took less than an hour even with some custom configurations.

2.2.3 Capabilities

Alfresco provides a content management system with file storage, meta-data storage, and search capabilities. File storage can be managed through a web-based interface, ftp interface, CIFS server (shared content appears as a shared drive in a Windows environment), NFS server (network file system for Linux/Unix systems), SharePoint, and other approaches. The system can be configured to maintain version control for every managed file. Hence, every new version of a file is retained by the system.

The system has the ability to perform some file conversion operations. For example, it can convert a Microsoft Word file to a PDF, and its advanced search can search the contents of some file types – not just the meta-data.

Alfresco has work flows that can assign tasks to people. It has a review work flow built-in, and custom work flows can be defined.

2.2.4 License

Alfresco is licensed under the Lesser Gnu Public License (LGPL) version 3. Details of their licensing are covered at this web page: http://wiki.alfresco.com/wiki/Open_Source_Licensing. They are willing to discuss providing Alfresco under different licensing terms including commercial or alternative free software licenses.

2.2.5 Assessment

Alfresco is a very mature, well-supported open-source project. It has both commercial support and community support through typical online systems. It provides end-user, developer and administrator documentation. It provides a sufficient backbone for the CCSI data management framework; we expect to be able to add connections between source files and output files corresponding to a particular run either through explicit relationships or using an additional metadata file generated during the simulation.

2.3 Apache Jackrabbit 2.6.1

2.3.1 Documentation & Web presence

Apache Jackrabbit is a fully conforming implementation of the Content Repository for Java Technology API (JSR 170 and 283). It is a hierarchical content store with versioning, transactions, and support for a variety of information types. Its specifications were developed and are part of the Java Community Process (<http://jcp.org/en/jsr/detail?id=170> and <http://jcp.org/en/jsr/detail?id=283>). The Apache Jackrabbit implementation of these specifications is covered by the <http://jackrabbit.apache.org/> website.

The Jackrabbit website provides comprehensive information and user support services. The documentation includes installation information, getting started information, and detailed developer documentation. The user/developer support includes mailing lists, an issue tracker, output of their Hudson continuous integration testing, and details on how to contribute back to Jackrabbit.

2.3.2 Installation

The downloads page, <http://jackrabbit.apache.org/downloads.html>, provides multiple ways to use Jackrabbit. I evaluated the Jackrabbit standalone JAR distribution which is simple to download and use with the OpenJDK 1.7.1 Java virtual machine. In this mode, it starts a webserver on port 8080.

2.3.3 Capabilities

The standalone Jackrabbit server provides two main methods access to the content repository. First, the content is accessible via the Web Distributed Authoring and Versioning (WebDAV) protocol. Second, the content can be accessed via the Java



Illustration 1: Accessing Apache Jackrabbit content repository via WebDAV protocol and the Chrome web browser.

Content Repository API (JCR API).

Using the WebDAV protocol, operating systems such as Windows, Linux, and Mac OS X can treat the content repository as a shared network filesystem. By making the content repository behave like a network filesystem, normal applications can access the content without specialized network interfaces. Clients can also access the content repository by a bare bones web interface.

The JCR API is a collection of Java interfaces to the content repository. The API can be used locally or remotely using remote method invocation (RMI). For local use, the CCSI code would need to be running in the same process as the Jackrabbit server, which seems impractical. If CCSI were to adopt Jackrabbit and use the Java API, we would primarily use RMI. In this approach, Java API commands are executed transparently across the network. For example, a Java program running on the end-user's desktop would initiate Java method calls that ultimately get executed on the Jackrabbit server.

2.3.4 License

Apache Jackrabbit releases are available under the Apache License, Version 2.0, see <http://jackrabbit.apache.org/downloads.html> and <http://www.apache.org/licenses/LICENSE-2.0>.

2.3.5 Assessment

Overall, Apache Jackrabbit seems like a solid system with a small set of fundamental features that enable it to work smoothly in a variety of environments. It doesn't support as many methods of file sharing as Alfresco; however, its WebDAV interface allows it to appear as a file system in most modern operating systems. It's worth noting that Alfresco also has a WebDAV interface and JCR API support.

There are several areas where Jackrabbit is less capable than Alfresco. Alfresco provides a rich web client to access and modify the content repository where Jackrabbit provides a minimalist WebDAV interface. Alfresco provides multi-language programming access to its content store via SOAP and the JCR API where Jackrabbit only supports languages other than Java through the WebDAV API. Jackrabbit is very Java focused, and Alfresco provides multiple access paths.

In comparison, Jackrabbit is a smaller and simpler system. Alfresco tends to support multiple ways of doing everything, and Jackrabbit supports one or two. This ultimately makes Alfresco a larger, more complex system, and Jackrabbit is in comparison smaller.

The support for Jackrabbit seems primarily community based. Based on the amount of web content, it appears to have a sufficient community to provide typical levels of open-source support.

2.4 HDF5 1.8.11

HDF5 is the Hierarchical Data Format 5, and it is the *de facto* file format standard for supercomputing data. It provides a high-performance library to read, write or modify HDF5 files in parallel onto a parallel file system. HDF5 is a very flexible, extensible file format, so frequently, communities will establish standards for HDF5 in their field. For example, the climate community established the NetCDF file format as their community standard, and starting with NetCDF-2, the NetCDF standard has used HDF as its underlying file format. Due to its role in the community, many tools have been created to process, analyze, and visualize HDF5 files.

2.4.1 Documentation & Web Presence

The HDF Group provides a website for HDF5, <http://www.hdfgroup.org/HDF5/>. They provide documentation for the basic HDF5 API, HDF5 tools, and the HDF5 specification. The HDF5 User's

Guide is 352 pages long, and the reference manual is 798 pages. There are also shorter tutorials to get users started using HDF5.

The HDF Group provides an email help desk help@hdfgroup.org. There are two mailing lists as well: one for announcements and one for community discussions. In addition, the HDF Group provides commercial support for those who wish to purchase it. They also provide training courses for a fee.

HDF5 installation packages are provided for Linux and Windows. For other systems, the source code is available. Most high-performance computing centers will have HDF5 pre-installed.

The HDF5 library can be accessed from multiple programming languages. The HDF Group provides documentation for the C, Fortran, C++, and Java interfaces to HDF5. There are third-party libraries to access HDF5 from Python (www.h5py.org).

2.4.2 Installation

Most major Linux distributions have precompiled HDF5 packages available. For example on a Red Hat system, you simply “yum install hdf5” or “yum install hdf5-openmpi”. Windows binaries or sources are available on the HDF Group site.

If you have a bizarre machine, you can download the HDF5 sources and build it yourself. It does use two 3rd party libraries which may also need to be installed by hand (gzip 2.1 and zlib).

2.4.3 Capabilities

HDF5 is an extensible file format and a library for accessing that file format. As its name suggests, the format is hierarchical – meaning that each file contains a directory structure of information. It is optimized for storing arrays, matrices, or high-dimensional arrays of integers and floating point numbers. Its design allows for significant meta-data. It can be used on serial or parallel computers, and many tools are written to support HDF5.

2.4.4 License

HDF5 is licensed under a BSD-style open-source license. Details of the licensing terms are covered here: <http://www.hdfgroup.org/HDF5/doc/Copyright.html>

2.4.5 Assessment

HDF5 is the community standard for large data storage, and it is the leading file format for HPC computing.

2.5 JSON

JSON (pronounced jay-sun) is a text-based file format for human- and machine-readable data interchange. Its name comes from JavaScript Object Notation. It is a simple yet powerful file format that can store hierarchical information. As its name suggests, JSON was originally design for the web; however, its simple, clear syntax led to it being adopted elsewhere. The CCSI Turbine project uses JSON for its configuration file format.

2.5.1 Documentation & Web Presence

JSON was originally documented by RFC-4627, and there is also a homepage for the file format, <http://www.json.org/>. There are libraries supporting JSON in numerous programming languages including C, C++, C#, Python, Java, JavaScript, and PHP. The homepage has links to these libraries and also JSON tools.

2.5.2 Installation

Most Linux distributions will have numerous JSON parsers prepackaged for installation. A JSON encoder and decoder is part of the Standard Python library since version 2.6.

2.5.3 Capabilities

JSON is a lightweight, hierarchical file format for data interchange. It can store information in an object-oriented way, and it also has support for array data. As a text-based file format, it is optimized for readability and interoperability rather than performance and compactness.

In many ways, it's similar to XML; however, it tends to be shorter than XML. The file format is also simpler than XML, so its parsers tend to be smaller and faster than corresponding XML parsers.

Like XML, JSON assumes that files are in Unicode, so it can store internationalized information. JSON is already used as the configuration file format for the CCSI Turbine code.

JSON Schema is a way of specifying what information a JSON file should contain. There are tools to verify that an JSON file validates against a particular JSON schema. A JSON Schema specification is itself a JSON file.

2.5.4 License

JSON is an open standard with multiple, independent implementations. For most languages, an open source parser implementation is available. In particular, since Python 2.6 a json parser has been available as part of the Python Standard Library (<http://www.python.org/download/releases/2.6.8/license/>). There are 24 implementations of json for Java, 11 for C++, and 16 for C.

2.5.5 Assessment

JSON is a simple, terse file format that balances the needs of human readability with simple parsing. It's a good choice for small data and configuration information. As a plaintext format, it's not optimized for large data or out-of-core processing.

2.6 XML

XML is a text-based file format that evolved as a generalization of HTML. Its initials come from eXtensible Markup Language, and it's meant to be human- and machine-readable. XML is meant to be a framework on which other file formats are build. For example, XHTML is an XML compatible version of HTML.

2.6.1 Documentation & Web Presence

XML has an excellent web presence. There are numerous sources for XML documentation, references, tutorials, and parsers. There are books on XML and books on uses of XML.

2.6.2 Installation

There are numerous XML parsers that are readily available and installable.

2.6.3 Capabilities

XML is an extensible, hierarchical file format. It's able to encode almost any type of information, and it is particular useful for data interchange between independently developed programs. XML is used for everything from exchanging order and inventory information to remote method invocation.

XML has two mechanisms for specifying what a valid document must contain. There is a Document Type Definition (DTD), and the XML Schema was introduced as a successor to DTDs. XML Schemas are the more powerful of the two specification languages.

XML is an international file format using Unicode as its character set.

2.6.4 License

XML is defined by the XML 1.0 Specification produced by the W3C. It is an open standard, and there are numerous open source XML parser implementations. XML parsers are available for practically every computer language. Apache Xerces provides C++, Java, and Perl XML parsers under the Apache public license v2.0 (<http://xerces.apache.org/xerces-c/index.html>). Python has included the expat parser as part of its standard library since Python 2.0.

2.6.5 Assessment

XML is a very popular standard in desktop and business computing. The main criticisms of XML are that it's verbose and complex. XML has a more elaborate syntax for opening and closing hierarchical elements which leads to XML files being long.

In comparison to JSON, the XML file format is much more complex. An XML parser, need to be able to parse XML and DTDs. This makes XML parsers more complex and makes XML parsing slower.

3 Recommendations

Based on the evaluation of these technologies, I recommend that CCSI adopt Alfresco as its data storage engine for the data management framework. It supports rich web client access, numerous file system access methods, and methods for programs to interact with the system. It's supported by a large open source community and a commercial entity.

For programs that do not already have a data format for their large-scale data, HDF5 is the recommended file format for data storage. We should adapt a common layout for CCSI-related HDF5 files, so we store data and meta-data in a consistent way.

We recognize that some programs have an internal, perhaps proprietary file format that is already supported. Our data management framework can support these files, but they will be opaque to the data management framework. We will not be able to do searching based on the file contents.

For configuration files, we recommend adopting JSON.