CINEMA

Cinema Database Specification
Dietrich Release v1.2

Document Version 1.1
(September 2018)
LA-UR-17-25072

by

David Rogers - dhr@lanl.gov
Jon Woodring - woodring@lanl.gov
James Ahrens - ahrens@lanl.gov
John Patchett - patchett@lanl.gov
Jonas Lukasczyk - jl@jluk.de

Los Alamos National Laboratory
Bikini Atoll Rd., SM 30
Los Alamos, NM 87545
cinema@lanl.gov

Contents

1 Introduction

This document is a specification for Cinema databases. It is specification D (Dietrich), version 1.3. This is a different approach than past specifications [1] [2], but is complimentary in spirit and function to them. This version is a simple embodiment of the new approach, which can be easily adapted to a wide range of use cases, and is designed for quick adoption by scientists, programmers and others.

See the Cinema website (`http://www.cinemascience.org`) for additional information, and contact the Cinema community (`cinema-info@lanl.gov`) or the authors of this document with questions.

2 Cinema Overview

Extreme scale scientific simulations are leading a charge to exascale computation, and data analytics runs the risk of being a bottleneck to scientific discovery. Due to power and I/O constraints, we expect in situ visualization and analysis will be a critical component of these workflows.

Options for extreme scale data analysis are often presented as a stark contrast: write large files to disk for interactive, exploratory analysis, or perform in situ analysis to save detailed data about phenomena that a scientist knows about in advance. Cinema represents a novel framework for a third option a highly interactive, data artifact-based approach that promotes exploration of simulation results, and is easily accessed through database specifications. This approach supports interactive exploration of a wide range of results, while still significantly reducing data movement and storage.

More information about the overall design of Cinema is available in the paper *An Image-based Approach to Extreme Scale In Situ Visualization and Analysis* [3].

A Cinema Database supports the following three use cases. Taken together, these support a novel method for interactively exploring artifacts from extremely large datasets.

1. Searching/querying of meta-data and data artifacts. Samples can be searched purely on metadata, content, position, time, or a combination of all of these.
2. Interactive visualization of sets of data artifacts.
3. Playing interactive visualizations, allowing the user on/off control of elements in the visualization.

2.1 What is a Cinema Database?

A Cinema database is a set of parameters mapped with a set of data artifacts. The artifacts can be images, grids, csv files - any type of data that can be written to disk.

A general design philosophy of Cinema is that applications reading and viewing a Cinema database can determine which data to operate on, and which operations to perform. This promotes a wide range of possible interactions with the data - not just the ones imagined by the creator of the database.

3 The Cinema D (Dietrich) v1.3 Specification

Cinema D (Dietrich) specification is a table-based data model for scientific data. In version 1.3 of this specification, we take the simplest approach to this new model, to simplify adoption and interaction with Cinema.

3.1 Required Elements

A Specification D version 1.3 Cinema database is directory that contains the following:

- a directory named `database_name.cdb` (*database_name* is the name of the database) containing:
  - `data.csv`, a Comma Separated Value file. This file is specified in Section 3.2.
  - optional data files, referenced by the `data.csv` file.
  - optional additional files and directories may be included in the directory, but are not part of this specification.

3.2 The `data.csv` File

The main data file for this specification is a Comma Separated Value file that has the following requirements. The last data column(s) may reference one or more external files. These referenced files can be of any file type or file extension. Note that new data can be added to or deleted from the `csv` file or files can be added to or removed from the database i.e., the `database_name.csv` directory, as long as the following requirements are met. The specification for the Spec D v1.3 `csv` follows (differing from v1.0 in points 5 and 9).

1. The file is UTF-8 encoded.
2. The file follows the specification rfc4180 [4], governing how fields are defined.
3. The first line of the file (header) is **required**.
   (a) Header values are unique non-empty **strings** that are the labels for the columns.
   (b) The last header values may be or start with `FILE`. No other non-`FILE` columns can occur after the first `FILE` columns (`FILE` columns are always last).
4. The remaining lines in the file are data rows.
   (a) There must be at least one data row.
   (b) Each data line is populated with an equal number of values (columns) as the header (i.e., the same number of separating commas per row).
   (c) All data values must be **floats** (floating-point values), **integers**, **strings**, or **empty** (missing) values.
   (d) All columns are **required** to have the same value type for all data rows, either **float**, **integer**, or **string**.
   (e) **Empty** values may appear in any column and data row, are represented by an empty string.
   (f) A literal empty **string** is represented by two double-quotes: `""`, which is different from both an **empty** (missing) value or **NaN**.
   (g) A data value representing a file path must be a string containing either a) a POSIX file path, or b) a URI.
      i. Files can be of any type, as indicated by MIME name extension, e.g., **.png**, **.vti**, **.txt**, etc.

3.2.1 Data Types

The first **non-empty** value is used to determine the data type of a column. We assume that built-in parsing in modern languages is available and can be used to test, parse, and convert values from the `csv` file.

In our previous examples **1** and **2**, the types of the columns are **integer**, four columns of **float**, and **string**. For the case of example **1**, determining the data types for columns 2 and 4 are deferred until the second row is parsed. All other subsequent values must be the same data type or **empty**.

Parsing precedence for determining the type of a column is: **integer**, **float**, **string**. That is, if the first non-empty value in a column parses as an **integer**, it is an **integer** column, otherwise if it parses as a **float**, it is a **float** column, otherwise it is a **string** column. *In particular, this means that floating-point values that are integers, must be written as floating point value: i.e., they must contain a decimal point.* For example, a value of 0 in a floating point column must be represented as 0.0.

Further explanation on particular values:

- `nan`, `NaN`, and `NAN` (or any combination of capitalization) are valid values, and should be handled by any implementation according to the type of the column, either **floats** or **strings**. If these are present in the *first* **non-empty** data row, then data type of the column should be **float** values. `NaN` is not a valid **integer** value for *any* row.
- Whitespace is not and should not be consumed (stripped) before and after comma separators. This is part of the rfc4180 specification. Whitespace *cannot* appear before or after a comma that has a double-quoted value, as that is an invalid rfc4180 file.
- Commas and double-quotes are special characters that require enclosing the value in double-quotes. A literal double-quote must be preceded by another double-quote: e.g., `"""` is the value `"`. (Please refer to rfc4180.)
- A literal empty string must be represented as `""`, as an empty string (no text between two commas, or before the first comma or after the last comma) is `NULL` (empty value). Whitespace between commas will be interpreted as the whitespace **not** as an empty string or empty value.

4 Contacts and Further Information

For further information, email the Cinema mailing list at `cinema-info@lanl.gov`, or contact the authors of this document. Additional information is available at the Cinema website, `http://www.cinemascience.org`.

5 Changes from v1.1

Version 1.2 supersedes version 1.1. All v1.1 files are compliant v1.2 files, but v1.2 files are not backwards compatible with v1.1. The following are the changes and clarifications that have been made from v1.1 to v1.2:

- The first data line can contain (`NULL`) values, which means that **empty** data values can occur anywhere.
- Clarifications made in section 3.2 under point 9.
- Clarifications made to section 3.2.2 on data types, especially related to the support of **empty** values.

6 Acknowledgements

The image used on the cover page is a publicity photo of Marlene Dietrich for the film Shanghai Express (1932). This photo is in the public domain [5].

7 Bibliography

References

[1] D. Rogers, J. Ahrens, and J. Patchett, "Cinema simple database specification," Tech. Rep. LA-UR-15-20572, Los Alamos National Laboratory, January 2015.

[2] D. Rogers, J. Woodring, J. Patchett, D. DeMarle, and B. Geveci, "Cinema database specification chaplin release," Tech. Rep. LA-UR-15-20645, Los Alamos National Laboratory, January 2017.

[3] J. Ahrens, S. Jourdain, P. O'Leary, J. Patchett, D. H. Rogers, and M. Petersen, "An image-based approach to extreme scale in situ visualization and analysis," in *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, SC '14, (Piscataway, NJ, USA), pp. 424–434, IEEE Press, 2014.

[4] Y. Shafranovich, "Common Format and MIME Type for Comma-Separated Values (CSV) Files." RFC 4180, Oct. 2005.

[5] Deanlaw, "Marlene dietrich in shanghai express (1932) by don english," 2016. `https://commons.wikimedia.org/wiki/File:Marlene_Dietrich_in_Shanghai_Express_(1932)_by_Don_English.png`.

A Cinema Spec D Cheat Sheet

A Specification D version 1.3 Cinema database is directory that contains the following:

- a directory named `database_name.cdb` (*database_name* is the name of the database) containing:
  - `data.csv`, a Comma Separated Value file. This file is specified in Section 3.2.
  - optional data files, referenced by the `data.csv` file.
  - optional additional files and directories may be included in the directory, but are not part of this specification.

A.1 Examples

Example 1 A Cinema database containing a table of data, but no references to data files.

```
database_name.cdb/
    data.csv
```

This `data.csv` file contains:

```
timestep,time value,x,y,z,category
1,0.3,1.0,1.1,1.2,one
2,0.2,2.0,2.1,2.2,two
```

Example 2 A Cinema database that includes metatdata, and references to local data files.

```
database_name.cdb/
    data.csv
    results_01.png
    results_02.png
```

This `data.csv` file contains:

```
timestep,time value,x,y,z,results
1,0.3,1.0,1.1,1.2,results_01.png
2,0.2,2.0,2.1,2.2,results_02.png
```

Example 3 A Cinema database that includes metatdata, and references to local data files not included in the Cinema database directory.

```
database_name.cdb/
    data.csv
```

This `data.csv` file contains:

```
timestep,time value,x,y,z,results
1,0.3,1.0,1.1,1.2,../../results-001/results_01.png
2,0.2,2.0,2.1,2.2,../../results-001/results_02.png
```

Example 4 A Cinema database that includes metatdata, and references to remote data files.

```
database_name.cdb/
    data.csv
```

This `data.csv` file contains:

```
timestep,time value,x,y,z,category,FILE
1,0.1,1.0,1.1,1.2,one,https://github.com/cinemascience/cinema/example_01.vti
2,0.2,2.0,2.1,2.2,two,https://github.com/cinemascience/cinema/example_02.vti
```