Apollo

Data storage specification

confidential

Revision number: 1

Date:

Name: P. van der Velde

Contents

[Introduction 2](#_Toc208928698)

[In-memory data storage 2](#_Toc208928699)

[File data storage 3](#_Toc208928700)

# Disclaimer

This specification is not, by any stretch of the imagination, complete. It will need to be revised several times before it is complete. Currently several major parts are either missing or incomplete. This disclaimer will be updated to reflect any change in these sections. Finally a specification is supposed to be a ‘living’ document and therefore never complete. What were you thinking, better learn to live with this fact.

# Introduction

There are two main data storage parts, the in-memory storage and the file storage. Both have different goals. The in-memory storage should keep the data that is currently being processed safe and consistent. The file storage should keep the project data safe while it is not being used. Data written to file should be complete and enough to restore the original project. Furthermore the file should be resistant against damage while still taking up the minimal amount of space.

# In-memory data storage

**Goal:**

To provide access by the user and the application to the data that is currently in use. The data storage should allow users to manipulate that data while keeping the data safe and consistent. The data storage should also allow users to get back to an earlier state if this is desired, in both small steps (undo) and large steps (snapshot). It is however not a goal to protect the data from unexpected events like failure of the application or even power failures. In the case of such an event data loss is acceptable.

DATA STORAGE

* Make sure data is always consistent. Can use transactions (commands) or immutable data for this. Immutable data has the benefit that we can also provide undo’s
* It should be possible to write time based data as a series (good for moving meshes, unsteady sims etc.)
* Should the data storage also store component info etc. or shall we separate that out into a different part?
* Allow data to be real data (data that can be used by components other than the generator) or ghost data (data that hasn’t really been committed, only usable by the generator and a visualizer). This is useful for suggested changes in e.g. the mesh generators etc.
* Data must have a state. Which can be:
  + Up to date
  + Invalidated (give info about why?)
  + Ghost (?)

# File data storage

* Store CRC’s etc. to make the file robust against corruption.