OPAT File Format Specification

Version 1.0

Emily M. Boudreaux

February 2025

Abstract

The untitled 4DSSE code (herafter "the code" will make use of the OPAT (Opacity Table) file format for efficient storage and retrieval of opacity tables parameterized in density and temperature. The format is designed to be extensible, self-contained, and computationally efficient, supporting fast lookups, structured metadata, and data integrity verification. All radiative opacity tables used by the code should be stored in OPAT format. Python bindings are provided to allow users to easily generate OPAT formated files.

1 Introduction

The OPAT format provides a structured binary format for storing opacity tables indexed by hydrogen mass fraction (X) and metal mass fraction (Z). It is designed for high-performance, data-integrity, and generalizability in mind. OPAT is the format that all radiative opacity tables are stored in for the code. The format ensures data integrity using SHA-256 checksums. All numeric values are stored as 64 bit floating point (double) values; this ensures approximately 15 decimals of precision. The file format is open source and full described in this document. Data are stored in **little endian** format.
Each OPAT file consists of three main sections:

1. **File Header**: Contains metadata, format version, and a lookup table offset.
2. **Data Tables**: Contains gridded opacity data.
3. **Table Index**: Maps (X, Z) values to byte offsets for direct access.

A few important things to note here are that the table index is stored at the **end** of the file. Therefore when reading, the index offset will need to be used in order to move the current position to the end of the file. This is done for ease of wrigint since the checksums of each table wont be known till the tables have been read in. Further, each table is intended to store opacity values as a function of log density ($\log R$) and log temperature ($\log T$). The actual numeric values in there could be anything, however, the more important takeaway is that each table stores its horizontal and vertical parameters as lists of length n and m and then its data as a list of length nxm.

2 File Structure

An OPAT file is structured as follows:

Section	Size (bytes)	Description
File Header	256 (fixed)	Metadata, format version, index offset
Data Tables	Variable	Opacity tables for different compositions
Table Index	Variable	Maps (X, Z) to byte locations

2.1 File Header

The header provides general metadata and file organization details.

Field	Type	Size (bytes)	Description
Magic Number	char[4]	4	"OPAT" identifier
Version	uint16	2	Format version (e.g., 1.0 = 0x0001)
Num Tables	uint32	4	Number of stored (X, Z) tables
Header Size	uint32	4	Byte offset to the Data Tables
Index Offset	uint64	8	Byte offset of Table Index
Creation Date	char[16]	16	Creation Date (Feb 15, 2025)
Source Info	char[64]	64	Description of data source
Comment	char[128]	128	Units, notes, etc.
Reserved	char[26]	26	Future use (zero-filled)

2.2 Table Index

Each entry in the index provides byte offsets for locating the corresponding opacity table.

Field	Type	Size (bytes)	Description
X Value	float64	8	Hydrogen mass fraction
Z Value	float64	8	Metal mass fraction
Byte Start	uint64	8	Start of data in file
Byte End	uint64	8	End of data in file
Checksum	char[32]	32	SHA-265 checksum of table

Each entry is 64 bytes long, allowing for efficient binary searching.

2.3 Data Tables

Each opacity table contains gridded values of opacity as a function of density and temperature.

Binary Storage Format:

```
struct OpacityTable {
    uint32_t N_R;    // Number of density points
    uint32_t N_T;    // Number of temperature points
    double* logR; // Log Density grid
    double* logT; // Log Temperature grid
    double* logK; // Log Opacity values
};
```

3 Checksum and Data Integrity

Each data table is assigned a SHA-256 checksum stored in the Table Index for validation.

4 Creation

Python scripts are provided to generate both mock data for testing as well as for ingesting various common opacity formats (OP, OPAL, OPLIB, Ferguson, Aesopus) into OPAT format. The python module opatio (in utils/opatio) provides a straightforward interface for the creation and reading of opat formated files. There is a readme in that directory which outlines its use.