ParticleZoo

ParticleZoo Reference Manual

v1.0.0

October 18, 2025

1 ParticleZoo 1

1.1 Overview . . . 2

1.2 Supported Formats . . . 3

1.3 Architecture . . . 4

1.4 Building and Installation . . . 5

1.5 Using the Library . . . 8

1.6 Command-Line Tools . . . 10

1.7 Extending the Library . . . 12

1.8 ROOT Format Support (Optional) . . . 13

1.9 Performance Considerations . . . 14

1.10 Troubleshooting . . . 15

2 License 17

3 Hierarchical Index 19

3.1 Class Hierarchy . . . 20

4 Class Index 21

4.1 Class List . . . 22

5 Namespace Documentation 25

5.1 ParticleZoo Namespace Reference . . . 26

6 Class Documentation 35

6.1 ParticleZoo::ByteBuffer Class Reference . . . 36

6.2 ParticleZoo::EGSphspFile::Reader Class Reference . . . 48

6.3 ParticleZoo::EGSphspFile::Writer Class Reference . . . 56

6.4 ParticleZoo::FixedValues Struct Reference . . . 64

6.5 ParticleZoo::FormatRegistry Class Reference . . . 66

6.6 ParticleZoo::IAEAphspFile::IAEAHeader Class Reference . . . 74

6.7 ParticleZoo::IAEAphspFile::Reader Class Reference . . . 109

6.8 ParticleZoo::IAEAphspFile::Writer Class Reference . . . 117

6.9 ParticleZoo::Particle Class Reference . . . 127

6.10 ParticleZoo::penEasyphspFile::Reader Class Reference . . . 145

6.11 ParticleZoo::penEasyphspFile::Writer Class Reference . . . 152

6.12 ParticleZoo::PhaseSpaceFileReader Class Reference . . . 158

6.13 ParticleZoo::PhaseSpaceFileWriter Class Reference 181
6.14 ParticleZoo::ROOT::BranchInfo Struct Reference 205
6.15 ParticleZoo::ROOT::Reader Class Reference 206
6.16 ParticleZoo::ROOT::Writer Class Reference 212
6.17 ParticleZoo::SupportedFormat Struct Reference 218
6.18 ParticleZoo::TOPASphspFile::Header Class Reference 219
6.19 ParticleZoo::TOPASphspFile::Header::DataColumn Struct Reference 228
6.20 ParticleZoo::TOPASphspFile::Header::ParticleStats Struct Reference 233
6.21 ParticleZoo::TOPASphspFile::Reader Class Reference 234
6.22 ParticleZoo::TOPASphspFile::Writer Class Reference 243
6.23 ParticleZoo::Version Struct Reference 253

Index **257**

Chapter 1

ParticleZoo

A high-performance C++20 library for reading, writing, and manipulating particle phase space files across multiple Monte Carlo simulation ecosystems. ParticleZoo provides a unified API that abstracts away format-specific details, enabling seamless interoperability between different simulation codes and workflows.

1.1 Overview

ParticleZoo serves as a universal translator and processor for particle phase space data, which represents the position, momentum, energy, and other properties of particles at specific locations in Monte Carlo simulations. The library is designed around a common `Particle` data model that can represent particles from any supported format, with automatic format detection and conversion capabilities.

Key Features

- **Unified API**: Single interface to work with multiple phase space formats
- **Format Transparency**: Automatic format detection with explicit override options
- **High Performance**: Efficient binary I/O with configurable buffering
- **Extensible Architecture**: Plugin-style registry system for adding new formats
- **Unit Consistency**: Built-in unit system ensures proper dimensional handling
- **Memory Efficient**: Streaming interfaces for processing large files
- **Cross-Platform**: Windows, Linux and macOS support with standard build tools

1.2 Supported Formats

The library includes built-in support for major Monte Carlo simulation formats:

- **EGS** (EGSnrc): `.egsphsp` files in MODE0 and MODE2, including suffixed variants (`.egsphsp1`, etc.)
- **IAEA**: `.IAEAphsp` International Atomic Energy Agency format with header files
- **TOPAS**: `.phsp` files in Binary, ASCII, and Limited variants
- **penEasy**: `.dat` ASCII format from the PENELOPE simulation code
- **ROOT** (optional): `.root` files generated with the CERN ROOT framework. Includes build-in templates for TOPAS and OpenGATE generated files. Also supports custom branch mappings

Additional formats can be added through the extensible registry system without modifying core library code.

1.3 Architecture

Core Components

Particle Class: The central data model representing a particle with position, momentum, energy, weight, and type information. Supports both standard properties and format-specific extensions through a flexible property system.

PhaseSpaceFileReader: Abstract base class for reading phase space files. Implementations handle format-specific parsing while presenting a common streaming interface.

PhaseSpaceFileWriter: Abstract base class for writing phase space files. Handles format-specific serialization with proper history counting.

FormatRegistry: Plugin-style system for registering and creating readers/writers. Enables runtime format discovery and automatic format detection from file extensions.

ByteBuffer: High-performance binary I/O buffer for efficient reading of large files with configurable buffering strategies.

Data Model

The `Particle` class provides access to:

- **Spatial coordinates**: Position (x, y, z) in configurable units
- **Momentum**: Direction cosines and kinetic energy
- **Particle properties**: PDG particle codes, statistical weight
- **History tracking**: Original history numbers and incremental counters
- **Format-specific data**: Extensible property system for specialized information

Unit System

ParticleZoo includes a comprehensive unit system that ensures dimensional consistency across different formats:

```
// Length units: mm, cm, m
float x_in_cm = particle.getX() / cm;
float y_in_mm = particle.getY() / mm;

// Energy units: eV, keV, MeV, GeV
float energy_MeV = particle.getKineticEnergy() / MeV;
```

1.4 Building and Installation

Prerequisites

- **Operating System**: Linux, macOS, or Windows
- **Compiler / Toolchain**:
  - Linux/macOS: C++20 compatible compiler (GCC 10+, Clang 13+)
  - Windows: Visual Studio 2019 or later with C++ development tools (MSVC)
- **Build Tools**:
  - Linux/macOS: GNU Make
  - Windows: Windows Command Prompt or PowerShell (uses `build.bat`)
- **Optional Dependencies**:
  - CERN ROOT (for ROOT format support)
    - Linux/macOS: requires `root-config` in PATH
    - Windows: not supported by build.bat script, requires manual compilation

Build Process

Linux and macOS

```
# Configure the build system (auto-detects compiler and dependencies)
./configure [--prefix=/your/installation/prefix]

# Build the library and tools
make          # Release build (default)
make debug    # Debug build with symbols
make release  # Explicitly build release version

# Install (optional)
make install  # defaults to /usr/local for the install PREFIX
make install PREFIX=/usr/local
```

Windows

```
# Configure, build, and optionally install
build.bat [--prefix=C:\path\to\install] [debug|release]
build.bat install [--prefix=C:\path\to\install] [debug|release]
```

Build Outputs

The build system creates the following artifacts:

Release build

- Linux/macOS: `build/gcc/release/`
- Windows: `build/msvc/release/`
  - Static library:
    - Linux/macOS: `libparticlezoo.a`
    - Windows: `libparticlezoo.lib`
  - Executables:
    - Linux/macOS: `PHSPConvert`, `PHSPCombine`, `PHSPImage`, `PHSPSplit`
    - Windows: `PHSPConvert.exe`, `PHSPCombine.exe`, `PHSPImage.exe`, `PHSPSplit.exe`
  - Dynamic library (Windows only): `build/msvc/release/bin/particlezoo.dll`

Debug build

- Linux/macOS: `build/gcc/debug/`
- Windows: `build/msvc/debug/`
  - Static library with debug symbols:
    - Linux/macOS: `libparticlezoo.a`
    - Windows: `libparticlezoo.lib`
  - Debug executables:
    - Linux/macOS: `PHSPConvert`, etc.
    - Windows: `PHSPConvert.exe`, etc.
  - Dynamic library (Windows only): `build/msvc/debug/bin/particlezoo.dll`

Installation (optional)

- Linux/macOS (with `make install`):
  - Headers: `$PREFIX/include/particlezoo/`
  - Static Library: `$PREFIX/lib/libparticlezoo.a`
  - Executables: `$PREFIX/bin/PHSPConvert`, etc.
- Windows (with `build.bat install`):
  - Headers: `PREFIX%\include\particlezoo\`
  - Static Library: `PREFIX%\lib\particlezoo.lib`
  - Executables and DLL: `PREFIX%\bin\PHSPConvert.exe`, etc.

Configuration Options

The `configure` script (Linux/macOS) accepts the following options:

- `--prefix=PATH` - Installation prefix (default: `/usr/local`)
- `--no-root` - Disable ROOT support even if available

The `build.bat` script (Windows) accepts the following options:

- `--prefix=PATH` - Installation prefix (default: `LOCALAPPDATA%`)

1.5 Using the Library

Basic Usage

Here's a simple example showing how to read from one format and write to another:

```
#include <particlezoo/PhaseSpaceFileReader.h>
#include <particlezoo/PhaseSpaceFileWriter.h>
#include <particlezoo/utilities/formats.h>

using namespace ParticleZoo;

int main() {
    // Register standard formats
    FormatRegistry::RegisterStandardFormats();

    // Create readers and writers - format auto-detected from extension
    auto reader = FormatRegistry::CreateReader("input.IAEAphsp");
    auto writer = FormatRegistry::CreateWriter("output.egsphsp");

    // Process all particles
    while (reader->hasMoreParticles()) {
        Particle particle = reader->getNextParticle();

        // Optionally modify particle properties
        // particle.setWeight(particle.getWeight() * 2.0);

        writer->writeParticle(particle);
    }

    // Clean up
    writer->close();
    reader->close();

    return 0;
}
```

Advanced Usage with Options

Many format readers and writers accept configuration options:

```
// Create options map for custom behavior
// Requires: #include <particlezoo/ROOT/ROOTphsp.h>
UserOptions options;
// Example: select predefined ROOT template when reading ROOT files
options[ParticleZoo::ROOT::ROOTFormatCommand] = { std::string("TOPAS") };

// Create reader with our user options
auto reader = FormatRegistry::CreateReader("ROOT", "simulation.root", options);
```

Advanced Usage with Fixed Values

Many formats support holding certain values (e.g. X, Y, Z) constant across all particles to reduce file sizes.

```
// Create default options map
UserOptions options;

// Create the flags for the fixed values and set the Z value to be constant at 100 cm
FixedValues fixedValues;
fixedValues.zIsConstant = true;
fixedValues.constantZ = 100 * cm;

// Create writer with explicit format, options, and a fixed Z value
auto writer = FormatRegistry::CreateWriter("IAEA", "simulation.IAEAphsp", options, fixedValues);
```

Working with Particles

The `Particle` class provides extensive access to particle properties:

```
Particle p = reader->getNextParticle();

// Basic properties
float x = p.getX() / cm;            // Position in cm
float y = p.getY() / cm;
float z = p.getZ() / cm;

float energy = p.getKineticEnergy() / MeV;  // Energy in MeV
float weight = p.getWeight();               // Statistical weight

// Direction (unit vector)
float dx = p.getDirectionalCosineX();
float dy = p.getDirectionalCosineY();
float dz = p.getDirectionalCosineZ();
```

Format-Specific Features

Different formats support different features. The library provides access to format-specific properties:

```
// EGS-specific latch information
if (p.hasIntProperty(IntPropertyType::EGS_LATCH)) {
    int latch = p.getIntProperty(IntPropertyType::EGS_LATCH);
}

// PENELOPE-specific interaction flags
if (p.hasIntProperty(IntPropertyType::PENELOPE_ILB1)) {
    int ilb1 = p.getIntProperty(IntPropertyType::PENELOPE_ILB1);
}
```

Error Handling

The library uses standard C++ exception handling:

```
try {
    auto reader = FormatRegistry::CreateReader("nonexistent.phsp");
    // ... process particles
} catch (const std::runtime_error& e) {
    std::cerr « "Error: " « e.what() « std::endl;
} catch (const std::exception& e) {
    std::cerr « "Unexpected error: " « e.what() « std::endl;
}
```

1.6 Command-Line Tools

ParticleZoo includes several command-line utilities that demonstrate the library's capabilities:

PHSPConvert - Format Conversion

Converts phase space files between different formats:

```
# Auto-detect formats from file extensions
PHSPConvert input.egsphsp output.IAEAphsp

# Explicitly specify formats
PHSPConvert --inputFormat EGS --outputFormat IAEA input.file output.file

# Limit particle count
PHSPConvert --maxParticles 1000000 input.IAEAphsp output.phsp

# Optional: project particles to a plane during conversion
PHSPConvert --projectToZ 100.0 input.phsp output.IAEAphsp
```

PHSPCombine - File Merging

Combines multiple phase space files into a single output:

```
# Combine multiple files
PHSPCombine --outputFile combined.IAEAphsp file1.egsphsp file2.egsphsp file3.egsphsp

# Mix formats during combination
PHSPCombine --outputFile result.phsp input1.IAEAphsp input2.egsphsp

# Preserve constant values in the output file if all input files have the same constant values
PHSPCombine --preserveConstants --outputFile result.IAEAphsp input1.IAEAphsp input2.IAEAphsp
```

PHSPImage - Visualization and Third Party Analysis

Creates 2D particle fluence or energy fluence images from phase space data. Can output either a detailed TIFF image with raw fluence data stored in 32-bit floats (default) which can be analyzed directly in third party tools like ImageJ, or in a simple bitmap BMP image with automatic constrast for easy visualization:

```
# Generate a flattened XY plane image (default)
PHSPImage beam.egsphsp fluence_map.tiff

# Generate project the particles to a specific XY plane (e.g. 100 cm or isocenter)
PHSPImage --projectTo 100 beam.egsphsp projection.tiff

# Custom plane and energy weighting, particles are not relocated, only particles located at
# Y = 5 cm +- a default margin of 0.25 cm will be counted (margin for XZ plane can be changed
# with the --tolerance parameter)
PHSPImage --outputFormat BMP --projectionType none --plane XZ --planeLocation 5.0 --energyWeighted
      simulation.IAEAphsp dose_profile.bmp
```

PHSPSplit - File Splitting

Splits a single phase space file into multiple (roughly) equally sized output files. History boundaries are respected, so individual files may differ slightly in size.

```
# Split a file into multiple parts
PHSPSplit --splitNumber 10 input.egsphsp

# Use short flag and specify output format
PHSPSplit -n 5 --outputFormat IAEA input.egsphsp
```

1.7 Extending the Library

Adding New Formats

To add support for a new phase space format:

1. **Implement Reader**: Inherit from `PhaseSpaceFileReader` and implement virtual methods
2. **Implement Writer**: Inherit from `PhaseSpaceFileWriter` and implement virtual methods
3. **Register Format**: Add registration call to connect file extensions with your implementations

Example registration:
```
SupportedFormat myFmt{"MyFormat", "My custom phase space format", ".myext"};
FormatRegistry::RegisterFormat(
    myFmt,
    [](const std::string& file, const UserOptions& opts) -> std::unique_ptr<PhaseSpaceFileReader> {
        return std::make_unique<MyFormatReader>(file, opts);
    },
    [](const std::string& file, const UserOptions& opts, const FixedValues & fixedValues) ->
      std::unique_ptr<PhaseSpaceFileWriter> {
        return std::make_unique<MyFormatWriter>(file, opts, fixedValues);
    }
);
```

Custom Particle Properties

The `Particle` class supports custom properties through the property system:
```
// Add custom integer property
particle.setIntProperty(IntPropertyType::CUSTOM, 42);

// Add custom float property
particle.setFloatProperty(FloatPropertyType::CUSTOM, 3.14f);

// Add custom boolean property
particle.setBoolProperty(BoolPropertyType::CUSTOM, true);
```

1.8 ROOT Format Support (Optional)

When compiled with ROOT support, ParticleZoo can read and write ROOT-based phase space files using predefined templates or custom branch mappings.

Predefined Templates

```
# Use TOPAS template
PHSPConvert --inputFormat ROOT --ROOT-format TOPAS input.root output.IAEAphsp

# Use OpenGATE template
PHSPConvert --inputFormat ROOT --ROOT-format OpenGATE simulation.root converted.egsphsp
```

Custom Branch Mapping

For ROOT files with non-standard branch names:

```
PHSPConvert --inputFormat ROOT \
    --ROOT-tree-name MyTree \
    --ROOT-energy E_kin \
    --ROOT-position-x pos_x \
    --ROOT-position-y pos_y \
    --ROOT-position-z pos_z \
    --ROOT-weight stat_weight \
    input.root output.phsp
```

Available branch mapping options:

- `--ROOT-tree-name <name>` - ROOT tree name
- `--ROOT-energy <branch>` - Energy branch
- `--ROOT-weight <branch>` - Statistical weight branch

- `--ROOT-position-x/y/z <branch>` - Position branches
- `--ROOT-cosine-x/y/z <branch>` - Direction cosines
- `--ROOT-cosine-z-sign <branch>` - Boolean flag for Z-direction sign
- `--ROOT-pdg-code <branch>` - Particle type identifier
- `--ROOT-history-number <branch>` - History counter

1.9 Performance Considerations

Memory Usage

- ParticleZoo uses streaming I/O to minimize memory footprint
- Configurable buffer sizes for optimal performance vs. memory trade-offs
- Large files can be processed with constant memory usage

Optimization Tips

- Use binary formats when possible for faster I/O
- Consider particle limits (`--maxParticles`) for testing and prototyping
- Enable compiler optimizations (`make release`) for production use
- ROOT format may be slower due to tree structure overhead

1.10 Troubleshooting

Common Issues

Build Problems:

- ∗"config.status not found"∗ → Run `./configure` before `make`
- ∗"The C++ standard in this build does not match ROOT configuration"∗ → The ROOT installation on your system was compiled with a different C++ standard than ParticleZoo. It may still work, but it cannot be guaranteed. Either rebuild both with the same C++ standard or use at your own risk.
- ∗"ROOT support: no"∗ → Ensure `root-config` is in PATH and re-run `./configure`
- ∗"checking whether g++ accepts -std=c++20... no"∗ → Update compiler (GCC 10+ or Clang 13+)

Runtime Issues:

- ∗"Unknown format"∗ → Use `--inputFormat` to explicitly specify format
- ∗"File not found"∗ → Check file paths and permissions

Performance Issues:

- Large files processing slowly → Consider using `--maxParticles` for testing
- Memory usage too high → Check if streaming interface is being used properly

Getting Help

For additional support:

1. Use `--formats` option to verify supported formats at runtime
2. Try explicit format specification with `--inputFormat`/`--outputFormat`
3. Verify file integrity using third-party format-specific validation tools if available

Chapter 2

License

Chapter 3

Hierarchical Index

3.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

ParticleZoo::ByteBuffer 36
ParticleZoo::FixedValues 64
ParticleZoo::FormatRegistry 66
ParticleZoo::IAEAphspFile::IAEAHeader 74
ParticleZoo::Particle 127
ParticleZoo::PhaseSpaceFileReader 158
    ParticleZoo::EGSphspFile::Reader 48
    ParticleZoo::IAEAphspFile::Reader 109
    ParticleZoo::ROOT::Reader 206
    ParticleZoo::TOPASphspFile::Reader 234
    ParticleZoo::penEasyphspFile::Reader 145
ParticleZoo::PhaseSpaceFileWriter 181
    ParticleZoo::EGSphspFile::Writer 56
    ParticleZoo::IAEAphspFile::Writer 117
    ParticleZoo::ROOT::Writer 212
    ParticleZoo::TOPASphspFile::Writer 243
    ParticleZoo::penEasyphspFile::Writer 152
ParticleZoo::ROOT::BranchInfo 205
ParticleZoo::SupportedFormat 218
ParticleZoo::TOPASphspFile::Header 219
ParticleZoo::TOPASphspFile::Header::DataColumn 228
ParticleZoo::TOPASphspFile::Header::ParticleStats 233
ParticleZoo::Version 253

Chapter 4

Class Index

4.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

ParticleZoo::ByteBuffer
Byte buffer used to improve I/O performance for reading and writing binary and text data 36
ParticleZoo::EGSphspFile::Reader
Reader class for EGS phase space files 48
ParticleZoo::EGSphspFile::Writer
Writer class for EGS phase space files 56
ParticleZoo::FixedValues
Structure defining constant (fixed) values for particle properties 64
ParticleZoo::FormatRegistry
Singleton registry for managing phase space file format readers and writers 66
ParticleZoo::IAEAphspFile::IAEAHeader
Header manager for IAEA phase space files 74
ParticleZoo::IAEAphspFile::Reader
Reader for IAEA format phase space files 109
ParticleZoo::IAEAphspFile::Writer
Writer for IAEA format phase space files 117
ParticleZoo::Particle
Represents a particle in phase space 127
ParticleZoo::penEasyphspFile::Reader
Reader for penEasy format phase space files 145
ParticleZoo::penEasyphspFile::Writer
Writer for penEasy format phase space files 152
ParticleZoo::PhaseSpaceFileReader
Base class for reading phase space files 158
ParticleZoo::PhaseSpaceFileWriter
Base class for writing phase space files 181
ParticleZoo::ROOT::BranchInfo
Configuration for ROOT TTree branch mapping 205
ParticleZoo::ROOT::Reader
ROOT format phase space file reader 206
ParticleZoo::ROOT::Writer
ROOT format phase space file writer 212
ParticleZoo::SupportedFormat
Structure describing a supported phase space file format 218
ParticleZoo::TOPASphspFile::Header
Header for TOPAS phase space files 219
ParticleZoo::TOPASphspFile::Header::DataColumn
Column definition for TOPAS phase space files 228
ParticleZoo::TOPASphspFile::Header::ParticleStats
Statistics tracking for individual particle types for TOPAS phase space files 233
ParticleZoo::TOPASphspFile::Reader
Reader for TOPAS phase space files 234

ParticleZoo::TOPASphspFile::Writer
Writer for TOPAS phase space files 243
ParticleZoo::Version
Version information and metadata for the ParticleZoo library 253

Chapter 5

Namespace Documentation

5.1 ParticleZoo Namespace Reference

Main namespace for the ParticleZoo phase space file processing library.

Classes

- class ByteBuffer

  Byte buffer used to improve I/O performance for reading and writing binary and text data.
- struct FixedValues

  Structure defining constant (fixed) values for particle properties.
- class FormatRegistry

  Singleton registry for managing phase space file format readers and writers.
- class Particle

  Represents a particle in phase space.
- class PhaseSpaceFileReader

  Base class for reading phase space files.
- class PhaseSpaceFileWriter

  Base class for writing phase space files.
- struct SupportedFormat

  Structure describing a supported phase space file format.
- struct Version

  Version information and metadata for the ParticleZoo library.

Typedefs

- using **byte** = unsigned char

  Type alias for unsigned byte (8 bits)
- using **signed_byte** = char

  Type alias for signed byte (8 bits)

Enumerations

- enum class BoolPropertyType { INVALID , IS_MULTIPLE_CROSSER , IS_SECONDARY_PARTICLE , CUSTOM }

  Enumeration of boolean property types for particles.
- enum class ByteOrder { LittleEndian = 1234 , BigEndian = 4321 , PDPEndian = 3412 }

  Enumeration of byte ordering schemes for multi-byte data types.
- enum class FloatPropertyType {
  INVALID , XLAST , YLAST , ZLAST ,
  CUSTOM }

Enumeration of floating-point property types for particles.

- enum class FormatType { BINARY , ASCII , NONE }

  Enumeration of file format types.
- enum class IntPropertyType {
  INVALID , INCREMENTAL_HISTORY_NUMBER , EGS_LATCH , PENELOPE_ILB1 ,
  PENELOPE_ILB2 , PENELOPE_ILB3 , PENELOPE_ILB4 , PENELOPE_ILB5 ,
  CUSTOM }

  Enumeration of integer property types for particles.
- enum class ParticleType : std::int32_t { }

  Strongly-typed enumeration of particle types with PDG codes.

Functions

- ParticleType getParticleTypeFromPDGID (std::int32_t pdg) noexcept

  Convert PDG identification code to ParticleType enumeration.
- constexpr std::string_view getParticleTypeName (ParticleType t)

  Get human-readable name for a particle type.
- std::int32_t getPDGIDFromParticleType (ParticleType type) noexcept

  Convert ParticleType enumeration to PDG identification code.
- std::ostream & operator<< (std::ostream &os, const ByteBuffer &buffer)

Variables

- constexpr std::size_t **DEFAULT_BUFFER_SIZE** = 1048576

  Default buffer size (1MiB)
- constexpr ByteOrder HOST_BYTE_ORDER

  The byte order of the host system.

5.1.1 Detailed Description

Main namespace for the ParticleZoo phase space file processing library.

ParticleZoo is a comprehensive C++ library for reading, writing, and manipulating phase space files from various Monte Carlo radiation transport codes. The library provides a unified interface for working with different file formats while preserving format-specific features and metadata.

The namespace contains all classes, functions, and utilities, including:

- Phase space file I/O operations
- Particle data structures and property management
- Universal interface for reading/writing to/from different phase space formats
- Format-specific readers and writers

Supported formats include EGS, IAEA, TOPAS, and others.

5.1.2 Enumeration Type Documentation

5.1.2.1 BoolPropertyType

```
enum class ParticleZoo::BoolPropertyType  [strong]
```

Enumeration of boolean property types for particles.

Defines standardized boolean flags that can be associated with particles from different Monte Carlo simulation codes.

Enumerator

INVALID	Invalid property type.
IS_MULTIPLE_CROSSER	Flag indicating that the particle crossed the phase space plane multiple times (assuming the phase space is planar)
IS_SECONDARY_PARTICLE	Flag indicating that the particle is a secondary.
CUSTOM	Custom boolean property type, can be used for any user-defined purpose.

5.1.2.2 ByteOrder

```
enum class ParticleZoo::ByteOrder  [strong]
```

Enumeration of byte ordering schemes for multi-byte data types.

Defines the different ways multi-byte values can be stored in memory, for cross-platform compatibility when reading/writing binary data files.

Enumerator

LittleEndian	Least significant byte first.
BigEndian	Most significant byte first.
PDPEndian	Mixed endian.

5.1.2.3 FloatPropertyType

```
enum class ParticleZoo::FloatPropertyType  [strong]
```

Enumeration of floating-point property types for particles.

Defines standardized float properties that can be associated with particles from different Monte Carlo simulation codes.

Enumerator

INVALID	Invalid property type, used for error checking.
XLAST	EGS-specific XLAST variable, for photons it is the X position of the last interaction, for electrons/positrons it is the X position it (or it's ancestor) was created at by a photon.
YLAST	EGS-specific YLAST variable, for photons it is the Y position of the last interaction, for electrons/positrons it is the Y position it (or it's ancestor) was created at by a photon.
ZLAST	EGS-specific ZLAST variable, for photons it is the Z position of the last interaction, for electrons/positrons it is the Z position it (or it's ancestor) was created at by a photon.
CUSTOM	Custom float property type, can be used for any user-defined purpose.

5.1.2.4 FormatType

```
enum class ParticleZoo::FormatType  [strong]
```

Enumeration of file format types.

Enumerator

BINARY	Binary format.
ASCII	ASCII text format.
NONE	Used for when ParticleZoo will not be responsible for reading/writing (e.g. ROOT)

5.1.2.5 IntPropertyType

```
enum class ParticleZoo::IntPropertyType  [strong]
```

Enumeration of integer property types for particles.

Defines standardized integer properties that can be associated with particles from different Monte Carlo simulation codes.

Enumerator

INVALID	Invalid property type, used for error checking.
INCREMENTAL_HISTORY_NUMBER	Sequential history number for tracking, tracks the number of new histories since the last particle was recorded.
EGS_LATCH	EGS-specific latch variable (see BEAMnrc User Manual, Chapter 8 for details)
PENELOPE_ILB1	PENELOPE ILB array value 1, corresponds to the generation of the particle (1 for primary, 2 for secondary, etc.)

Enumerator

PENELOPE_ILB2	PENELOPE ILB array value 2, corresponds to the particle type of the particle's parent (applies only if ILB1 $>$ 1)
PENELOPE_ILB3	PENELOPE ILB array value 3, corresponds to the interaction type that created the particle (applies only if ILB1 $>$ 1)
PENELOPE_ILB4	PENELOPE ILB array value 4, is non-zero if the particle is created by atomic relaxation and corresponds to the atomic transistion that created the particle.
PENELOPE_ILB5	PENELOPE ILB array value 5, a user-defined value which is passed on to all descendant particles created by this particle.
CUSTOM	Custom integer property type, can be used for any user-defined purpose.

5.1.2.6 ParticleType

```
enum class ParticleZoo::ParticleType :  std::int32_t  [strong]
```

Strongly-typed enumeration of particle types with PDG codes.

This enum class provides type-safe access to PDG particle codes while maintaining the standardized integer values. Each enumerator corresponds to a specific particle type with its official PDG identification number.

Special values internal to ParticleZoo (these codes should not be written to files):

- Unsupported (99): For particle types that are not supported by ParticleZoo
- PseudoParticle (98): For pseudo-particles containing simulation metadata

The enum uses std::int32_t as the underlying type to match PDG code specifications and handle the full range of positive and negative values.

Note

Generated automatically from PARTICLE_LIST macro

Enumerator

Unsupported	Unknown or non-standard particle type.
PseudoParticle	Simulation-specific pseudo-particle.

5.1.3 Function Documentation

5.1.3.1 getParticleTypeFromPDGID()

```
ParticleType ParticleZoo::getParticleTypeFromPDGID (
             std::int32_t pdg )  [inline], [noexcept]
```

Convert PDG identification code to ParticleType enumeration.

Performs efficient lookup from standardized PDG integer codes to the corresponding strongly-typed ParticleType enumeration value. This function provides the primary interface for particle identification in Monte Carlo simulation data processing.

Parameters

pdg	The PDG identification code (standardized integer)

Returns

ParticleType enumeration value, or ParticleType::Unsupported for unknown codes

Note

Uses compile-time generated switch statement for O(1) lookup performance

5.1.3.2 getParticleTypeName()

```
constexpr std::string_view ParticleZoo::getParticleTypeName (
             ParticleType t )  [constexpr]
```

Get human-readable name for a particle type.

Returns the string representation of a ParticleType enumeration value, providing descriptive names for particles in logging, debugging, and user interface contexts.

Parameters

t	The ParticleType enumeration value

Returns

std::string_view containing the particle name (compile-time constant)

Note

Marked constexpr for compile-time evaluation

Returns string_view for efficiency

5.1.3.3 getPDGIDFromParticleType()

```
std::int32_t ParticleZoo::getPDGIDFromParticleType (
            ParticleType type )  [inline], [noexcept]
```

Convert ParticleType enumeration to PDG identification code.

Extracts the standardized PDG integer code from a ParticleType enumeration value. This provides the inverse operation to getParticleTypeFromPDGID(), enabling conversion from strongly-typed enums back to the integer codes required by some phase space formats.

Parameters

type	The ParticleType enumeration value

Returns

std::int32_t PDG identification code

5.1.3.4 operator<<()

```
std::ostream & ParticleZoo::operator<< (
            std::ostream & os,
            const ByteBuffer & buffer )  [inline]
```

Parameters

os	The output stream to write to
buffer	The ByteBuffer to write from

Returns

std::ostream& Reference to the output stream for chaining

5.1.4 Variable Documentation

5.1.4.1 HOST_BYTE_ORDER

```
constexpr ByteOrder ParticleZoo::HOST_BYTE_ORDER  [constexpr]
```

Initial value:

```
=
    (std::endian::native == std::endian::little) ? ByteOrder::LittleEndian :
    (std::endian::native == std::endian::big)    ? ByteOrder::BigEndian :
                                                   ByteOrder::PDPEndian
```

The byte order of the host system.

Automatically determined at compile time based on the system's native byte order.

Chapter 6

Class Documentation

6.1 ParticleZoo::ByteBuffer Class Reference

Byte buffer used to improve I/O performance for reading and writing binary and text data.

```
#include <particlezoo/ByteBuffer.h>
```

Public Member Functions

- ByteBuffer (const std::span< const byte > data, ByteOrder byteOrder=HOST_BYTE_ORDER)

  Create a ByteBuffer from a span of existing data.
- ByteBuffer (std::size_t bufferSize=DEFAULT_BUFFER_SIZE, ByteOrder byteOrder=HOST_BYTE_ORDER)

  Create an empty ByteBuffer with a fixed capacity.
- std::size_t appendData (ByteBuffer &buffer, bool ignoreOffset=false)

  Append data from another ByteBuffer to this buffer.
- std::size_t appendData (std::istream &stream)

  Append data from an input stream to the existing buffer content.
- std::size_t capacity () const

  Get the total capacity of the buffer.
- void **clear** ()

  Reset the buffer, resetting the offset and length to 0.
- void compact ()

  Compact the buffer by moving unread data to the beginning.
- const byte ∗ data () const

  Get a pointer to the raw buffer data.
- void expand ()

  Expand the buffer to its full capacity, filling unused space with zeros.
- ByteOrder getByteOrder () const

  Get the current byte order setting.
- std::size_t length () const

  Get the length of valid data in the buffer.
- void moveTo (std::size_t offset)

  Move the read/write offset to a specific position in the buffer.
- template< typename T >
  T read ()

  Read a primitive type T from the buffer with automatic byte order conversion.
- std::span< const byte > readBytes (std::size_t len)

  Read a span of bytes from the buffer.
- std::string readLine ()

  Read a line of ASCII text from the buffer.
- std::string readString ()

  Read a null-terminated string from the buffer.

- std::string readString (std::size_t stringLength)

  Read a string of specified length from the buffer.
- std::size_t remainingToRead () const

  Get the number of bytes remaining to be read from current offset.
- std::size_t remainingToWrite () const

  Get the number of bytes available for writing.
- void setByteOrder (ByteOrder byteOrder)

  Set the byte order for interpreting multi-byte data types.
- std::size_t setData (std::istream &stream)

  Initialize the buffer with data from an input stream.
- std::size_t setData (std::span< const byte > data)

  Initialize the buffer with data from a span.
- template<typename T >
  void write (const T &value)

  Write a primitive type T to the buffer with automatic byte order conversion.
- void writeBytes (std::span< const byte > data)

  Write a span of bytes to the buffer.
- void writeString (const std::string &str, bool includeNullTerminator=false)

  Write a string to the buffer.

6.1.1 Detailed Description

Byte buffer used to improve I/O performance for reading and writing binary and text data.

ByteBuffer provides efficient buffered I/O operations with automatic byte order conversion for cross-platform compatibility. It supports both reading from and writing to the buffer, with automatic capacity management and various data type read/write operations.

The buffer maintains both a current offset (read/write position) and a length (amount of valid data), allowing for flexible positioning and partial reads/writes.

6.1.2 Constructor & Destructor Documentation

6.1.2.1 ByteBuffer() [1/2]

```
ParticleZoo::ByteBuffer::ByteBuffer (
              std::size_t bufferSize = DEFAULT_BUFFER_SIZE,
              ByteOrder byteOrder = HOST_BYTE_ORDER )  [inline]
```

Create an empty ByteBuffer with a fixed capacity.

Note

The capacity must be greater than zero.

Parameters

bufferSize	The maximum capacity of the buffer in bytes (default: DEFAULT_BUFFER_SIZE)
byteOrder	The byte order for multi-byte data types (default: HOST_BYTE_ORDER)

Exceptions

std::runtime_error	if bufferSize is zero

6.1.2.2 ByteBuffer() [2/2]

```
ParticleZoo::ByteBuffer::ByteBuffer (
             const std::span< const byte > data,
             ByteOrder byteOrder = HOST_BYTE_ORDER )  [inline]
```

Create a ByteBuffer from a span of existing data.

The buffer is initialized with a copy of the provided data.

Parameters

data	A span containing the initial data to copy into the buffer
byteOrder	The byte order for multi-byte data types (default: HOST_BYTE_ORDER)

6.1.3 Member Function Documentation

6.1.3.1 appendData() [1/2]

```
std::size_t ParticleZoo::ByteBuffer::appendData (
             ByteBuffer & buffer,
             bool ignoreOffset = false )  [inline]
```

Append data from another ByteBuffer to this buffer.

Copies data from the source buffer and appends it after the current data.

Parameters

buffer	The source ByteBuffer to copy data from
ignoreOffset	If true, copies all data from source; if false, only unread data

Returns

std::size_t The number of bytes appended

Exceptions

std::runtime_error	if the combined data exceeds buffer capacity

6.1.3.2 appendData() `[2/2]`

```
std::size_t ParticleZoo::ByteBuffer::appendData (
            std::istream & stream )  [inline]
```

Append data from an input stream to the existing buffer content.

Reads additional data from the stream and appends it after the current data. Does not modify the current offset.

Parameters

stream	The input stream to read from

Returns

std::size_t The number of bytes appended from the stream

Exceptions

std::runtime_error	if buffer is full or no data could be read

6.1.3.3 capacity()

```
std::size_t ParticleZoo::ByteBuffer::capacity ( ) const  [inline]
```

Get the total capacity of the buffer.

Returns

std::size_t The maximum number of bytes the buffer can hold

6.1.3.4 compact()

```
void ParticleZoo::ByteBuffer::compact ( )  [inline]
```

Compact the buffer by moving unread data to the beginning.

Shifts any unread data (from current offset to end) to the start of the buffer and updates the length and offset accordingly. Useful for reclaiming space after partial reads.

6.1.3.5 data()

```
const byte * ParticleZoo::ByteBuffer::data ( ) const  [inline]
```

Get a pointer to the raw buffer data.

Returns

const byte* Pointer to the beginning of the buffer data

6.1.3.6 expand()

```
void ParticleZoo::ByteBuffer::expand ( )  [inline]
```

Expand the buffer to its full capacity, filling unused space with zeros.

Extends the data length to the full buffer capacity by filling the remaining space with zero-ed bytes.

6.1.3.7 getByteOrder()

```
ByteOrder ParticleZoo::ByteBuffer::getByteOrder ( ) const  [inline]
```

Get the current byte order setting.

Returns

ByteOrder The byte order used for multi-byte data types

6.1.3.8 length()

```
std::size_t ParticleZoo::ByteBuffer::length ( ) const  [inline]
```

Get the length of valid data in the buffer.

Returns

std::size_t The number of bytes of valid data

6.1.3.9 moveTo()

```
void ParticleZoo::ByteBuffer::moveTo (
             std::size_t offset )  [inline]
```

Move the read/write offset to a specific position in the buffer.

Parameters

offset	The new offset position (must be <= current data length)

Exceptions

std::runtime_error	if offset exceeds the current data length

6.1.3.10 read()

```
template<typename T >
T ParticleZoo::ByteBuffer::read ( )  [inline]
```

Read a primitive type T from the buffer with automatic byte order conversion.

Reads sizeof(T) bytes from the current offset and converts the byte order if necessary. Advances the offset by sizeof(T).

Template Parameters

T	The primitive type to read (must be trivially copyable)

Returns

T The value read from the buffer

Exceptions

std::runtime_error	if insufficient data is available

6.1.3.11 readBytes()

```
std::span< const byte > ParticleZoo::ByteBuffer::readBytes (
            std::size_t len )  [inline]
```

Read a span of bytes from the buffer.

Returns a view of the requested bytes without copying. Advances the offset by the requested length.

Parameters

len	The number of bytes to read

Returns

std::span<const byte> A span view of the requested bytes

Exceptions

std::runtime_error	if insufficient data is available

6.1.3.12 readLine()

```
std::string ParticleZoo::ByteBuffer::readLine ( )  [inline]
```

Read a line of ASCII text from the buffer.

Reads characters until a newline ('
') is found. Automatically handles Windows-style line endings by removing trailing '\r'. Advances the offset past the newline.

Returns

std::string The line read from the buffer (without newline characters)

Exceptions

std::runtime_error	if newline is not found or no data is available

6.1.3.13 readString() [1/2]

```
std::string ParticleZoo::ByteBuffer::readString ( )  [inline]
```

Read a null-terminated string from the buffer.

Reads characters until a null terminator ('\0') is found. Advances the offset past the null terminator.

Returns

std::string The string read from the buffer (without null terminator)

Exceptions

std::runtime_error	if null terminator is not found or insufficient data

6.1.3.14 readString() [2/2]

```
std::string ParticleZoo::ByteBuffer::readString (
             std::size_t stringLength )  [inline]
```

Read a string of specified length from the buffer.

Reads exactly the specified number of characters. Advances the offset by the string length.

Parameters

stringLength	The number of characters to read

Returns

std::string The string read from the buffer

Exceptions

std::runtime_error	if insufficient data is available

6.1.3.15 remainingToRead()

```
std::size_t ParticleZoo::ByteBuffer::remainingToRead ( ) const  [inline]
```

Get the number of bytes remaining to be read from current offset.

Returns

std::size_t The number of unread bytes (length - offset)

6.1.3.16 remainingToWrite()

```
std::size_t ParticleZoo::ByteBuffer::remainingToWrite ( ) const  [inline]
```

Get the number of bytes available for writing.

Returns

std::size_t The remaining capacity (capacity - length)

6.1.3.17 setByteOrder()

```
void ParticleZoo::ByteBuffer::setByteOrder (
            ByteOrder byteOrder )  [inline]
```

Set the byte order for interpreting multi-byte data types.

Parameters

byteOrder	The byte order to use for subsequent read/write operations

6.1.3.18 setData() [1/2]

```
std::size_t ParticleZoo::ByteBuffer::setData (
              std::istream & stream )  [inline]
```

Initialize the buffer with data from an input stream.

Reads up to the buffer's capacity from the stream. Replaces any existing data. Resets the offset to 0.

Parameters

stream	The input stream to read from

Returns

std::size_t The number of bytes read from the stream

Exceptions

std::runtime_error	if no data could be read from the stream

6.1.3.19 setData() [2/2]

```
std::size_t ParticleZoo::ByteBuffer::setData (
              std::span< const byte > data )  [inline]
```

Initialize the buffer with data from a span.

Replaces any existing data in the buffer. Resets the offset to 0.

Parameters

data	A span containing the data to copy into the buffer

Returns

std::size_t The number of bytes copied

Exceptions

std::runtime_error	if data size exceeds buffer capacity

6.1.3.20 write()

```
template<typename T >
void ParticleZoo::ByteBuffer::write (
            const T & value )  [inline]
```

Write a primitive type T to the buffer with automatic byte order conversion.

Converts the value to the buffer's byte order if necessary and writes sizeof(T) bytes. Advances the offset by sizeof(T) and updates the length if necessary.

Template Parameters

T	The primitive type to write (must be trivially copyable)

Parameters

value	The value to write to the buffer

Exceptions

std::runtime_error	if insufficient space is available

6.1.3.21 writeBytes()

```
void ParticleZoo::ByteBuffer::writeBytes (
            std::span< const byte > data )  [inline]
```

Write a span of bytes to the buffer.

Copies the bytes from the span to the buffer. Advances the offset by the data size and updates the length if necessary.

Parameters

data	A span containing the bytes to write

Exceptions

std::runtime_error	if insufficient space is available

6.1.3.22 writeString()

```
void ParticleZoo::ByteBuffer::writeString (
             const std::string & str,
             bool includeNullTerminator = false )  [inline]
```

Write a string to the buffer.

Writes the string's characters with or without (default) a null terminator. Advances the offset by the string length and updates the length if necessary.

Parameters

str	The string to write to the buffer
includeNullTerminator	If true, appends a null terminator after the string

Exceptions

std::runtime_error	if insufficient space is available

The documentation for this class was generated from the following file:

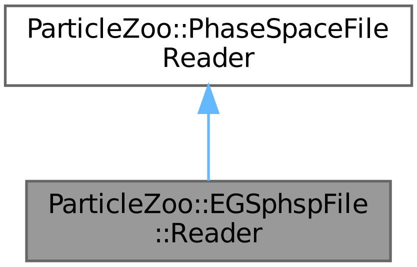
- include/particlezoo/ByteBuffer.h

6.2 ParticleZoo::EGSphspFile::Reader Class Reference

Reader class for EGS phase space files.

```
#include <particlezoo/egs/egsphspFile.h>
```

Inheritance diagram for ParticleZoo::EGSphspFile::Reader:


ParticleZoo::PhaseSpaceFile Reader
ParticleZoo::EGSphspFile ::Reader

Public Member Functions

- Reader (const std::string &fileName, const UserOptions &options=UserOptions{})

    Construct a new EGS phase space file reader.
- float getMaxKineticEnergy () const

    Get the maximum kinetic energy of particles in the file.
- float getMinElectronEnergy () const

    Get the minimum electron energy threshold used in the simulation.
- EGSMODE getMode () const

    Get the EGS file mode (MODE0 or MODE2).
- std::uint64_t getNumberOfOriginalHistories () const override

    Get the number of original Monte Carlo histories.
- std::uint64_t getNumberOfParticles () const override

    Get the total number of particles in the phase space file.
- unsigned int getNumberOfPhotons () const

    Get the number of photons in the phase space.

Public Member Functions inherited from ParticleZoo::PhaseSpaceFileReader

- PhaseSpaceFileReader (const std::string &phspFormat, const std::string &fileName, const UserOptions &user↩ Options, FormatType formatType=FormatType::BINARY, const FixedValues fixedValues=FixedValues(), unsigned int bufferSize=DEFAULT_BUFFER_SIZE)

  Construct a new Phase Space File Reader object.
- virtual ∼PhaseSpaceFileReader ()

  Destroy the Phase Space File Reader object.
- void close ()

  Close the phase space file and clean up resources.
- float getConstantPx () const

  Get the constant X-component of the direction unit vector (if constant).
- float getConstantPy () const

  Get the constant Y-component of the direction unit vector (if constant).
- float getConstantPz () const

  Get the constant Z-component of the direction unit vector (if constant).
- float getConstantWeight () const

  Get the constant statistical weight value (if constant).
- float getConstantX () const

  Get the constant X coordinate value (if constant).
- float getConstantY () const

  Get the constant Y coordinate value (if constant).
- float getConstantZ () const

  Get the constant Z coordinate value (if constant).
- const std::string getFileName () const

  Get the filename of the phase space file being read.
- std::uint64_t getFileSize () const

  Get the size of the phase space file in bytes.
- const FixedValues getFixedValues () const

  Get the fixed values configuration.
- virtual std::uint64_t getHistoriesRead ()

  Get the number of Monte Carlo histories that have been read so far. If the end of the file has been reached, this will return the total number of original histories unless more histories than expected have already been read - in which case it returns the actual count.
- Particle getNextParticle ()

  Get the next particle from the phase space file.
- virtual std::uint64_t getParticlesRead ()

  Get the number of particles that have been read so far.
- const std::string getPHSPFormat () const

  Get the phase space file format identifier.
- virtual bool hasMoreParticles ()

  Check if there are more particles to read in the file.

- bool isPxConstant () const

  Check if the X-component of momentum is constant for all particles.
- bool isPyConstant () const

  Check if the Y-component of momentum is constant for all particles.
- bool isPzConstant () const

  Check if the Z-component of momentum is constant for all particles.
- bool isWeightConstant () const

  Check if the statistical weight is constant for all particles.
- bool isXConstant () const

  Check if the X coordinate is constant for all particles.
- bool isYConstant () const

  Check if the Y coordinate is constant for all particles.
- bool isZConstant () const

  Check if the Z coordinate is constant for all particles.
- void moveToParticle (std::uint64_t particleIndex)

  Move the file position to a specific particle index.
- void setCommentMarkers (const std::vector< std::string > &commentMarkers)

  Set comment markers for ASCII format files.

Static Public Member Functions

- static std::vector< CLICommand > getFormatSpecificCLICommands ()

  Get the list of EGS-specific command line interface commands.

Static Public Member Functions inherited from ParticleZoo::PhaseSpaceFileReader

- static std::vector< CLICommand > getCLICommands ()

  Get command line interface commands supported by this reader.

Protected Member Functions

- std::size_t getParticleRecordLength () const override

  Get the length of each particle record in bytes.
- std::size_t getParticleRecordStartOffset () const override

  Get the byte offset where particle records start.
- Particle readBinaryParticle (ByteBuffer &buffer) override

  Read a single particle in EGS binary format.

Protected Member Functions inherited from ParticleZoo::PhaseSpaceFileReader

- double calcThirdUnitComponent (double &u, double &v) const

  Calculate the third component of a unit vector from two components (double precision).
- float calcThirdUnitComponent (float &u, float &v) const

  Calculate the third component of a unit vector from two components (float precision).
- const ByteBuffer getHeaderData ()

  Get the file header data as a byte buffer.
- const ByteBuffer getHeaderData (std::size_t headerSize)

  Get a specific amount of header data as a byte buffer.
- virtual std::size_t getMaximumASCIILineLength () const

  Get the maximum line length for ASCII format files.
- Particle getNextParticle (bool countParticleInStatistics)

  Get the next particle with optional statistics counting control.
- std::size_t getNumberOfEntriesInFile () const

  Get the number of particle records that fit in the file.
- virtual std::uint64_t getParticlesRead (bool includeSkippedParticles)

  Get the number of particles read with optional inclusion of skipped particles (including pseudo-particles).
- const UserOptions & getUserOptions () const

  Get the user options that were passed to the constructor.
- virtual Particle readASCIIParticle (const std::string &line)

  Read a particle from ASCII data.
- virtual Particle readParticleManually ()

  Read a particle manually (for formats requiring third-party I/O).
- void setByteOrder (ByteOrder byteOrder)

  Set the byte order for binary data interpretation.
- void setConstantPx (float Px)

  Set a constant X-component of the direction unit vector for all particles.
- void setConstantPy (float Py)

  Set a constant Y-component of the direction unit vector for all particles.
- void setConstantPz (float Pz)

  Set a constant Z-component of the direction unit vector for all particles.
- void setConstantWeight (float weight)

  Set a constant statistical weight for all particles.
- void setConstantX (float X)

  Set a constant X coordinate value for all particles.
- void setConstantY (float Y)

  Set a constant Y coordinate value for all particles.
- void setConstantZ (float Z)

  Set a constant Z coordinate value for all particles.

6.2.1 Detailed Description

Reader class for EGS phase space files.

Provides functionality to read EGS phase space files created by EGSnrc Monte Carlo simulations (or it's variants BEAMnrc, DOSXYZnrc, etc.). Supports both MODE0 and MODE2 formats.

6.2.2 Constructor & Destructor Documentation

6.2.2.1 Reader()

```
ParticleZoo::EGSphspFile::Reader::Reader (
            const std::string & fileName,
            const UserOptions & options = UserOptions{} )
```

Construct a new EGS phase space file reader.

Parameters

fileName	Path to the EGS phase space file to read
options	User options including user-specific configuration

Exceptions

std::runtime_error	if file format is invalid or unsupported mode

6.2.3 Member Function Documentation

6.2.3.1 getFormatSpecificCLICommands()

```
std::vector< CLICommand > ParticleZoo::EGSphspFile::Reader::getFormatSpecificCLICommands ( )
[static]
```

Get the list of EGS-specific command line interface commands.

Returns

std::vector<CLICommand> Vector of EGS-specific CLI commands

6.2.3.2 getMaxKineticEnergy()

`float ParticleZoo::EGSphspFile::Reader::getMaxKineticEnergy ( ) const [inline]`

Get the maximum kinetic energy of particles in the file.

Returns

float Maximum kinetic energy

6.2.3.3 getMinElectronEnergy()

`float ParticleZoo::EGSphspFile::Reader::getMinElectronEnergy ( ) const [inline]`

Get the minimum electron energy threshold used in the simulation.

Returns

float Minimum electron energy

6.2.3.4 getMode()

`EGSMODE ParticleZoo::EGSphspFile::Reader::getMode ( ) const [inline]`

Get the EGS file mode (MODE0 or MODE2).

Returns

EGSMODE The detected file mode

6.2.3.5 getNumberOfOriginalHistories()

`std::uint64_t ParticleZoo::EGSphspFile::Reader::getNumberOfOriginalHistories ( ) const [inline], [override], [virtual]`

Get the number of original Monte Carlo histories.

Returns

std::uint64_t Number of original histories that generated this phase space

Implements ParticleZoo::PhaseSpaceFileReader.

6.2.3.6 getNumberOfParticles()

```
std::uint64_t ParticleZoo::EGSphspFile::Reader::getNumberOfParticles ( ) const  [inline], [override],
[virtual]
```

Get the total number of particles in the phase space file.

Returns

std::uint64_t Number of particles (from header or calculated from file size)

Implements ParticleZoo::PhaseSpaceFileReader.

6.2.3.7 getNumberOfPhotons()

```
unsigned int ParticleZoo::EGSphspFile::Reader::getNumberOfPhotons ( ) const  [inline]
```

Get the number of photons in the phase space.

Returns

unsigned int Number of photon particles

6.2.3.8 getParticleRecordLength()

```
std::size_t ParticleZoo::EGSphspFile::Reader::getParticleRecordLength ( ) const  [inline], [override],
[protected], [virtual]
```

Get the length of each particle record in bytes.

Returns

std::size_t Record length (28 for MODE0, 32 for MODE2)

Reimplemented from ParticleZoo::PhaseSpaceFileReader.

6.2.3.9 getParticleRecordStartOffset()

```
std::size_t ParticleZoo::EGSphspFile::Reader::getParticleRecordStartOffset ( ) const  [inline],
[override], [protected], [virtual]
```

Get the byte offset where particle records start.

Returns

std::size_t Offset from the beginning of the file (same as record length for EGS files)

Reimplemented from ParticleZoo::PhaseSpaceFileReader.

6.2.3.10 readBinaryParticle()

```
Particle ParticleZoo::EGSphspFile::Reader::readBinaryParticle (
             ByteBuffer & buffer )  [override], [protected], [virtual]
```

Read a single particle in EGS binary format.

Parses EGS binary format including LATCH bits and ZLAST if present.

Parameters

buffer	The byte buffer containing particle data

Returns

Particle The parsed particle object with EGS-specific properties

Exceptions

std::runtime_error	if particle data is invalid

Reimplemented from ParticleZoo::PhaseSpaceFileReader.

The documentation for this class was generated from the following files:

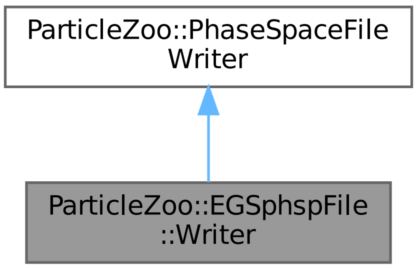
- include/particlezoo/egs/egsphspFile.h
- src/egs/egsphspFile.cc

6.3 ParticleZoo::EGSphspFile::Writer Class Reference

Writer class for EGS phase space files.

```
#include <particlezoo/egs/egsphspFile.h>
```

Inheritance diagram for ParticleZoo::EGSphspFile::Writer:


ParticleZoo::PhaseSpaceFile Writer
ParticleZoo::EGSphspFile ::Writer

Public Member Functions

- Writer (const std::string &fileName, const UserOptions &options=UserOptions{})

  Construct a new EGS phase space file writer.
- std::uint64_t getMaximumSupportedParticles () const override

  Get the maximum number of particles this format can support.
- void setNumberOfOriginalHistories (unsigned int numberOfOriginalHistories)

  Manually set the number of original Monte Carlo histories.

Public Member Functions inherited from ParticleZoo::PhaseSpaceFileWriter

- PhaseSpaceFileWriter (const std::string &phspFormat, const std::string &fileName, const UserOptions &user↩ Options, FormatType formatType=FormatType::BINARY, const FixedValues fixedValues=FixedValues(), unsigned int bufferSize=DEFAULT_BUFFER_SIZE)

  Construct a new Phase Space File Writer object.
- virtual ∼PhaseSpaceFileWriter ()

  Destroy the Phase Space File Writer object.
- void addAdditionalHistories (std::uint64_t additionalHistories)

Add additional Monte Carlo histories to the count.

- void close ()

  Close the phase space file and finalize writing.
- ByteOrder getByteOrder () const

  Get the byte order used for binary data writing.
- float getConstantPx () const

  Get the constant X-component of the direction unit vector (if constant).
- float getConstantPy () const

  Get the constant Y-component of the direction unit vector (if constant).
- float getConstantPz () const

  Get the constant Z-component of the direction unit vector (if constant).
- float getConstantWeight () const

  Get the constant statistical weight value (if constant).
- float getConstantX () const

  Get the constant X coordinate value (if constant).
- float getConstantY () const

  Get the constant Y coordinate value (if constant).
- float getConstantZ () const

  Get the constant Z coordinate value (if constant).
- const std::string getFileName () const

  Get the filename where the phase space file is being written.
- const FixedValues getFixedValues () const

  Get the fixed values configuration.
- virtual std::uint64_t getHistoriesWritten () const

  Get the number of Monte Carlo histories that have been written.
- std::uint64_t getParticlesWritten () const

  Get the number of particles that have been written to the file.
- const std::string getPHSPFormat () const

  Get the phase space file format identifier.
- bool isPxConstant () const

  Check if the X-component of the direction unit vector is set to a constant value for all particles.
- bool isPyConstant () const

  Check if the Y-component of the direction unit vector is set to a constant value for all particles.
- bool isPzConstant () const

  Check if the Z-component of the direction unit vector is set to a constant value for all particles.
- bool isWeightConstant () const

  Check if the statistical weight is set to a constant value for all particles.
- bool isXConstant () const

  Check if the X coordinate is set to a constant value for all particles.
- bool isYConstant () const

  Check if the Y coordinate is set to a constant value for all particles.

- bool isZConstant () const

  Check if the Z coordinate is set to a constant value for all particles.
- virtual void writeParticle (Particle particle)

  Write a particle to the phase space file.

Static Public Member Functions

- static std::vector< CLICommand > getFormatSpecificCLICommands ()

  Get the list of EGS-specific command line interface commands.

Static Public Member Functions inherited from ParticleZoo::PhaseSpaceFileWriter

- static std::vector< CLICommand > getCLICommands ()

  Get command line interface commands supported by this writer.

Protected Member Functions

- std::size_t getParticleRecordLength () const override

  Get the length of each particle record in bytes.
- std::size_t getParticleRecordStartOffset () const override

  Get the byte offset where particle records start.
- virtual void writeBinaryParticle (ByteBuffer &buffer, Particle &particle) override

  Write a single particle in EGS binary format.
- virtual void writeHeaderData (ByteBuffer &buffer) override

  Write the EGS file header with current statistics.

Protected Member Functions inherited from ParticleZoo::PhaseSpaceFileWriter

- virtual bool accountForAdditionalHistories (std::uint64_t additionalHistories)

  Handle accounting for simulation histories that produced no particles.
- virtual bool canHaveConstantPx () const

  Check if this format supports constant X-component of the direction unit vector.
- virtual bool canHaveConstantPy () const

  Check if this format supports constant Y-component of the direction unit vector.
- virtual bool canHaveConstantPz () const

  Check if this format supports constant Z-component of the direction unit vector.
- virtual bool canHaveConstantWeight () const

  Check if this format supports constant statistical weights.
- virtual bool canHaveConstantX () const

Check if this format supports constant X coordinates.
- virtual bool canHaveConstantY () const

  Check if this format supports constant Y coordinates.
- virtual bool canHaveConstantZ () const

  Check if this format supports constant Z coordinates.
- virtual bool canWritePseudoParticlesExplicitly () const

  Check if this format can write pseudo-particles explicitly.
- virtual void fixedValuesHaveChanged ()

  Called when fixed values have been changed.
- virtual size_t getMaximumASCIILineLength () const

  Get the maximum line length for ASCII format files.
- virtual std::uint64_t getPendingHistories () const

  Get the number of pending histories to account for.
- const UserOptions & getUserOptions () const

  Get the user options that were passed to the constructor.
- void setByteOrder (ByteOrder byteOrder)

  Set the byte order for binary data writing.
- void setConstantPx (float Px)

  Set a constant X-component of the direction unit vector for all particles.
- void setConstantPy (float Py)

  Set a constant Y-component of the direction unit vector for all particles.
- void setConstantPz (float Pz)

  Set a constant Z-component of the direction unit vector for all particles.
- void setConstantWeight (float weight)

  Set a constant statistical weight for all particles.
- void setConstantX (float X)

  Set a constant X coordinate value for all particles.
- void setConstantY (float Y)

  Set a constant Y coordinate value for all particles.
- void setConstantZ (float Z)

  Set a constant Z coordinate value for all particles.
- virtual const std::string writeASCIIParticle (Particle &particle)

  Write a particle in ASCII format as a string.
- virtual void writeParticleManually (Particle &particle)

  Write a particle manually (for formats requiring third-party I/O).

6.3.1 Detailed Description

Writer class for EGS phase space files.

Provides functionality to write EGS phase space files compatible with EGSnrc Monte Carlo simulations (or it's variants BEAMnrc, DOSXYZnrc, etc.). Supports both MODE0 and MODE2 formats.

6.3.2 Constructor & Destructor Documentation

6.3.2.1 Writer()

```
ParticleZoo::EGSphspFile::Writer::Writer (
             const std::string & fileName,
             const UserOptions & options = UserOptions{} )
```

Construct a new EGS phase space file writer.

Parameters

fileName	Path where the EGS phase space file will be written
options	User options including EGS-specific configuration (e.g., mode selection)

Exceptions

std::runtime_error	if specified mode is unsupported

6.3.3 Member Function Documentation

6.3.3.1 getFormatSpecificCLICommands()

```
std::vector< CLICommand > ParticleZoo::EGSphspFile::Writer::getFormatSpecificCLICommands ( )
[static]
```

Get the list of EGS-specific command line interface commands.

Returns

std::vector<CLICommand> Vector of EGS-specific CLI commands for writers

6.3.3.2 getMaximumSupportedParticles()

```
std::uint64_t ParticleZoo::EGSphspFile::Writer::getMaximumSupportedParticles ( ) const  [inline],
[override], [virtual]
```

Get the maximum number of particles this format can support.

Returns

std::uint64_t Maximum particle count (limited by 32-bit unsigned integer)

Implements ParticleZoo::PhaseSpaceFileWriter.

6.3.3.3 getParticleRecordLength()

```
std::size_t ParticleZoo::EGSphspFile::Writer::getParticleRecordLength ( ) const  [inline], [override],
[protected], [virtual]
```

Get the length of each particle record in bytes.

Returns

std::size_t Record length (28 for MODE0, 32 for MODE2)

Reimplemented from ParticleZoo::PhaseSpaceFileWriter.

6.3.3.4 getParticleRecordStartOffset()

```
std::size_t ParticleZoo::EGSphspFile::Writer::getParticleRecordStartOffset ( ) const  [inline],
[override], [protected], [virtual]
```

Get the byte offset where particle records start.

Returns

std::size_t Offset from the beginning of the file (same as record length for EGS files)

Reimplemented from ParticleZoo::PhaseSpaceFileWriter.

6.3.3.5 setNumberOfOriginalHistories()

```
void ParticleZoo::EGSphspFile::Writer::setNumberOfOriginalHistories (
             unsigned int numberOfOriginalHistories )  [inline]
```

Manually set the number of original Monte Carlo histories.

Allows explicit specification of the history count instead of automatic tracking.

Parameters

numberOfOriginalHistories	The number of original histories to record in the header

6.3.3.6 writeBinaryParticle()

```
void ParticleZoo::EGSphspFile::Writer::writeBinaryParticle (
            ByteBuffer & buffer,
            Particle & particle )  [override], [protected], [virtual]
```

Write a single particle in EGS binary format.

Converts particle data to EGS format including LATCH encoding and ZLAST if in MODE2.

Parameters

buffer	The byte buffer to write particle data into
particle	The particle to write

Exceptions

std::runtime_error	if particle type is unsupported or required properties are missing

Reimplemented from ParticleZoo::PhaseSpaceFileWriter.

6.3.3.7 writeHeaderData()

```
void ParticleZoo::EGSphspFile::Writer::writeHeaderData (
            ByteBuffer & buffer )  [override], [protected], [virtual]
```

Write the EGS file header with current statistics.

Writes the header containing mode string, particle counts, energy ranges, and history information based on accumulated statistics (or specified histories).

Parameters

buffer	The byte buffer to write header data into

Exceptions

std::runtime_error	if mode is unsupported

Implements ParticleZoo::PhaseSpaceFileWriter.

6.3 ParticleZoo::EGSphspFile::Writer Class Reference

The documentation for this class was generated from the following files:

- include/particlezoo/egs/egsphspFile.h
- src/egs/egsphspFile.cc

6.4 ParticleZoo::FixedValues Struct Reference

Structure defining constant (fixed) values for particle properties.

```
#include <particlezoo/Particle.h>
```

Public Member Functions

- bool operator== (const FixedValues &other) const

  Equality comparison operator for FixedValues.

Public Attributes

- float **constantPx** {0}

  Constant X directional cosine value (when pxIsConstant is true)
- float **constantPy** {0}

  Constant Y directional cosine value (when pyIsConstant is true)
- float **constantPz** {0}

  Constant Z directional cosine value (when pzIsConstant is true)
- float **constantWeight** {1}

  Constant statistical weight value (when weightIsConstant is true)
- float **constantX** {0}

  Constant X coordinate value (when xIsConstant is true)
- float **constantY** {0}

  Constant Y coordinate value (when yIsConstant is true)
- float **constantZ** {0}

  Constant Z coordinate value (when zIsConstant is true)
- bool **pxIsConstant** {0}

  True if X directional cosine is constant for all particles.
- bool **pyIsConstant** {0}

  True if Y directional cosine is constant for all particles.
- bool **pzIsConstant** {0}

  True if Z directional cosine is constant for all particles.
- bool **weightIsConstant** {0}

  True if statistical weight is constant for all particles.
- bool **xIsConstant** {0}

  True if X coordinate is constant for all particles.
- bool **yIsConstant** {0}

  True if Y coordinate is constant for all particles.
- bool **zIsConstant** {0}

  True if Z coordinate is constant for all particles.

6.4.1 Detailed Description

Structure defining constant (fixed) values for particle properties.

Used to optimize phase space files by storing constant values once rather than repeating them for every particle. Useful when all particles share certain properties (e.g., all particles start from the same position).

6.4.2 Member Function Documentation

6.4.2.1 operator==()

```
bool ParticleZoo::FixedValues::operator== (
             const FixedValues & other ) const  [inline]
```

Equality comparison operator for FixedValues.

Parameters

other	The other FixedValues object to compare with

Returns

true if all members are equal

false if any members differ

The documentation for this struct was generated from the following file:

- include/particlezoo/Particle.h

6.5 ParticleZoo::FormatRegistry Class Reference

Singleton registry for managing phase space file format readers and writers.

```
#include <particlezoo/utilities/formats.h>
```

Public Types

- using ReaderFactoryFn = std::function< std::unique_ptr< PhaseSpaceFileReader >(const std::string &filename, const UserOptions &options)>

  Function type for creating phase space file readers.
- using WriterFactoryFn = std::function< std::unique_ptr< PhaseSpaceFileWriter >(const std::string &filename, const UserOptions &options, const FixedValues &fixedValues)>

  Function type for creating phase space file writers.

Static Public Member Functions

- static std::unique_ptr< PhaseSpaceFileReader > CreateReader (const std::string &filename, const UserOptions &options={})

  Create a reader for a file using automatic format detection.
- static std::unique_ptr< PhaseSpaceFileReader > CreateReader (const std::string &formatName, const std::string &filename, const UserOptions &options={})

  Create a reader for a specific format and file.
- static std::unique_ptr< PhaseSpaceFileWriter > CreateWriter (const std::string &filename, const UserOptions &options={}, const FixedValues &fixedValues={})

  Create a writer for a file using automatic format detection.
- static std::unique_ptr< PhaseSpaceFileWriter > CreateWriter (const std::string &formatName, const std::string &filename, const UserOptions &options={}, const FixedValues &fixedValues={})

  Create a writer for a specific format and file.
- static const std::string ExtensionForFormat (const std::string &formatName)

  Get the standard file extension for a specific format.
- static std::vector< SupportedFormat > FormatsForExtension (const std::string &extension)

  Find all formats that support a given file extension.
- static void PrintSupportedFormats ()

  Print a list of all supported formats to standard output.
- static void RegisterFormat (const SupportedFormat &fmt, ReaderFactoryFn reader, WriterFactoryFn writer)

  Register a new phase space file format with reader and writer factories.
- static void RegisterStandardFormats ()

  Register all standard built-in phase space file formats.
- static const std::vector< SupportedFormat > SupportedFormats ()

  Get a list of all registered formats.

Static Public Attributes

- static constexpr bool FILE_EXTENSION_CAN_HAVE_SUFFIX = true

  Constant indicating that a file extension can have numeric suffixes.

6.5.1 Detailed Description

Singleton registry for managing phase space file format readers and writers.

The FormatRegistry provides a centralized thread-safe system for registering, discovering, and creating phase space file readers and writers for different simulation formats. It supports automatic format detection based on file extensions and provides factory methods for creating appropriate reader/writer instances.

The registry is typically populated during application startup by calling RegisterStandardFormats(), which registers all built-in format handlers.

6.5.2 Member Typedef Documentation

6.5.2.1 ReaderFactoryFn

```
using ParticleZoo::FormatRegistry::ReaderFactoryFn = std::function<std::unique_ptr<PhaseSpaceFileReader>(const
std::string& filename, const UserOptions& options)>
```

Function type for creating phase space file readers.

Parameters

filename	The path to the file to read
options	User options for configuring the reader

Returns

std::unique_ptr<PhaseSpaceFileReader> A unique pointer to the created reader

6.5.2.2 WriterFactoryFn

```
using ParticleZoo::FormatRegistry::WriterFactoryFn = std::function<std::unique_ptr<PhaseSpaceFileWriter>(const
std::string& filename, const UserOptions& options, const FixedValues& fixedValues)>
```

Function type for creating phase space file writers.

Parameters

filename	The path to the file to write
options	User options for configuring the writer
fixedValues	Fixed values for constant particle properties

Returns

std::unique_ptr<PhaseSpaceFileWriter> A unique pointer to the created writer

6.5.3 Member Function Documentation

6.5.3.1 CreateReader() `[1/2]`

```
std::unique_ptr< PhaseSpaceFileReader > ParticleZoo::FormatRegistry::CreateReader (
            const std::string & filename,
            const UserOptions & options = {} )  [static]
```

Create a reader for a file using automatic format detection.

Determines the appropriate format based on the file extension and creates a reader instance. Requires a unique format match for the extension.

Parameters

filename	The path to the file to read (must have a recognized extension)
options	User options for configuring the reader (default: empty)

Returns

std::unique_ptr<PhaseSpaceFileReader> A unique pointer to the created reader

Exceptions

std::runtime_error	if no extension found, no format matches, or multiple formats match

6.5.3.2 CreateReader() [2/2]

```
std::unique_ptr< PhaseSpaceFileReader > ParticleZoo::FormatRegistry::CreateReader (
            const std::string & formatName,
            const std::string & filename,
            const UserOptions & options = {} )  [static]
```

Create a reader for a specific format and file.

Creates a reader instance for the specified format, bypassing automatic detection.

Parameters

formatName	The name of the format to use (must be registered)
filename	The path to the file to read
options	User options for configuring the reader (default: empty)

Returns

std::unique_ptr<PhaseSpaceFileReader> A unique pointer to the created reader

Exceptions

std::runtime_error	if the format is not registered

6.5.3.3 CreateWriter() [1/2]

```
std::unique_ptr< PhaseSpaceFileWriter > ParticleZoo::FormatRegistry::CreateWriter (
            const std::string & filename,
            const UserOptions & options = {},
            const FixedValues & fixedValues = {} )  [static]
```

Create a writer for a file using automatic format detection.

Determines the appropriate format based on the file extension and creates a writer instance. Requires a unique format match for the extension.

Parameters

filename	The path to the file to write (must have a recognized extension)
options	User options for configuring the writer (default: empty)
fixedValues	Fixed values for constant particle properties (default: empty)

Returns

std::unique_ptr<PhaseSpaceFileWriter> A unique pointer to the created writer

Exceptions

std::runtime_error	if no extension found, no format matches, or multiple formats match

6.5.3.4 CreateWriter() [2/2]

```
std::unique_ptr< PhaseSpaceFileWriter > ParticleZoo::FormatRegistry::CreateWriter (
             const std::string & formatName,
             const std::string & filename,
             const UserOptions & options = {},
             const FixedValues & fixedValues = {} )  [static]
```

Create a writer for a specific format and file.

Creates a writer instance for the specified format, bypassing automatic detection.

Parameters

formatName	The name of the format to use (must be registered)
filename	The path to the file to write
options	User options for configuring the writer (default: empty)
fixedValues	Fixed values for constant particle properties (default: empty)

Returns

std::unique_ptr<PhaseSpaceFileWriter> A unique pointer to the created writer

Exceptions

std::runtime_error	if the format is not registered

6.5.3.5 ExtensionForFormat()

```
const std::string ParticleZoo::FormatRegistry::ExtensionForFormat (
             const std::string & formatName )  [static]
```

Get the standard file extension for a specific format.

Parameters

formatName	The name of the format to query

Returns

const std::string The standard file extension for the format

Exceptions

std::runtime_error	if the format is not registered

6.5.3.6 FormatsForExtension()

```
std::vector< SupportedFormat > ParticleZoo::FormatRegistry::FormatsForExtension (
            const std::string & extension )  [static]
```

Find all formats that support a given file extension.

Performs case-insensitive matching of file extensions. Also handles formats that support extensions with numeric suffixes (e.g., ".egsphsp1" matching ".egsphsp").

Parameters

extension	The file extension to search for (including the dot)

Returns

std::vector<SupportedFormat> Vector of formats supporting the extension

6.5.3.7 PrintSupportedFormats()

```
void ParticleZoo::FormatRegistry::PrintSupportedFormats ( )  [static]
```

Print a list of all supported formats to standard output.

Displays format names, descriptions, and file extensions in a human-readable format.

6.5.3.8 RegisterFormat()

```
void ParticleZoo::FormatRegistry::RegisterFormat (
             const SupportedFormat & fmt,
             ReaderFactoryFn reader,
             WriterFactoryFn writer )  [static]
```

Register a new phase space file format with reader and writer factories.

Registers a format with the global registry, making it available for automatic detection and creation. The format name must be unique.

Parameters

fmt	The format metadata (name, description, extension, etc.)
reader	Factory function for creating readers of this format
writer	Factory function for creating writers of this format

Exceptions

std::invalid_argument	if format parameters are invalid
std::runtime_error	if format name is already registered

6.5.3.9 RegisterStandardFormats()

```
void ParticleZoo::FormatRegistry::RegisterStandardFormats ( )  [static]
```

Register all standard built-in phase space file formats.

This method registers the standard formats supported by ParticleZoo including:

- IAEA format (.IAEAphsp)
- TOPAS format (.phsp)
- penEasy format (.dat)
- EGS format (.egsphsp with suffixes)
- ROOT format (.root) - if compiled with ROOT support

This method is safe to call multiple times (uses internal flag to prevent duplicate registration). Should be called during application initialization before using the registry.

6.5.3.10 SupportedFormats()

```
const std::vector< SupportedFormat > ParticleZoo::FormatRegistry::SupportedFormats ( )  [static]
```

Get a list of all registered formats.

Returns a copy of all currently registered format metadata.

Returns

const std::vector<SupportedFormat> Vector of all supported formats

6.5.4 Member Data Documentation

6.5.4.1 FILE_EXTENSION_CAN_HAVE_SUFFIX

```
constexpr bool ParticleZoo::FormatRegistry::FILE_EXTENSION_CAN_HAVE_SUFFIX = true  [static], [constexpr]
```

Constant indicating that a file extension can have numeric suffixes.

Used when registering formats that support numbered file extensions (e.g., ".egsphsp1", ".egsphsp2").

The documentation for this class was generated from the following files:

- include/particlezoo/utilities/formats.h
- src/utilities/formats.cc

6.6 ParticleZoo::IAEAphspFile::IAEAHeader Class Reference

Header manager for IAEA phase space files.

```
#include <particlezoo/IAEA/IAEAHeader.h>
```

Public Types

- enum class EXTRA_FLOAT_TYPE { USER_DEFINED_GENERIC_TYPE = 0 , XLAST = 1 , YLAST = 2 , ZLAST = 3 }

    Extra float data types for IAEA format.
- enum class EXTRA_LONG_TYPE {
  USER_DEFINED_GENERIC_TYPE = 0 , INCREMENTAL_HISTORY_NUMBER = 1 , EGS_LATCH = 2 , PENELOPE_ILB5 = 3 ,
  PENELOPE_ILB4 = 4 , PENELOPE_ILB3 = 5 , PENELOPE_ILB2 = 6 , PENELOPE_ILB1 = 7 }

    Extra integer data types for IAEA format.
- enum class FileType { PHSP_FILE = 0 , PHSP_GENERATOR = 1 }

    File type classification for IAEA phase space files.
- enum class SECTION {
  IAEA_INDEX , TITLE , FILE_TYPE , CHECKSUM ,
  RECORD_CONTENTS , RECORD_CONSTANT , RECORD_LENGTH , BYTE_ORDER ,
  ORIGINAL_HISTORIES , PARTICLES , PHOTONS , ELECTRONS ,
  POSITRONS , NEUTRONS , PROTONS , TRANSPORT_PARAMETERS ,
  MACHINE_TYPE , MONTE_CARLO_CODE_VERSION , GLOBAL_PHOTON_ENERGY_CUTOFF , GLOBAL_PARTICLE_ENERG
  ,
  COORDINATE_SYSTEM_DESCRIPTION , BEAM_NAME , FIELD_SIZE , NOMINAL_SSD ,
  MC_INPUT_FILENAME , VARIANCE_REDUCTION_TECHNIQUES , INITIAL_SOURCE_DESCRIPTION , PUBLISHED_REFERENCE ,
  AUTHORS , INSTITUTION , LINK_VALIDATION , ADDITIONAL_NOTES ,
  STATISTICAL_INFORMATION_PARTICLES , STATISTICAL_INFORMATION_GEOMETRY , CUSTOM_SECTION
  }

    Header section identifiers for IAEA format.

Public Member Functions

- IAEAHeader (const IAEAHeader &other, const std::string &newFilePath)

    Copy constructor with new file path.
- IAEAHeader (const std::string &filePath, bool newFile=false)

    Construct header from existing IAEA header file.
- void addExtraFloat (EXTRA_FLOAT_TYPE type)

    Add an extra float data type to the record format.
- void addExtraLong (EXTRA_LONG_TYPE type)

Add an extra integer data type to the record format.

- bool checksumIsValid () const

  Validate the data integrity checksum This check is strict. Not only does it verify that the checksum matches the file size, but it also checks that it equals the record length multiplied by the number of particles.
- void countParticleStats (const Particle &particle)

  Update particle statistics with a new particle.
- ByteOrder getByteOrder () const

  Get the byte order for binary data (endianness)
- std::uint64_t getChecksum () const

  Get the data integrity checksum.
- float getConstantU () const

  Get the constant U direction cosine value (when not stored per particle)
- float getConstantV () const

  Get the constant V direction cosine value (when not stored per particle)
- float getConstantW () const

  Get the constant W direction cosine value (when not implicitly stored per particle)
- float getConstantWeight () const

  Get the constant particle weight value (when not stored per particle)
- float getConstantX () const

  Get the constant X coordinate value (when not stored per particle)
- float getConstantY () const

  Get the constant Y coordinate value (when not stored per particle)
- float getConstantZ () const

  Get the constant Z coordinate value (when not stored per particle)
- std::string getDataFilePath () const

  Get the path to the associated data file.
- EXTRA_FLOAT_TYPE getExtraFloatType (unsigned int index) const

  Get the type of the extra float value at the specified index.
- EXTRA_LONG_TYPE getExtraLongType (unsigned int index) const

  Get the type of the extra integer value at the specified index.
- FileType getFileType () const

  Get the file type classification.
- std::string getHeaderFilePath () const

  Get the path to the header file.
- std::string getIAEAIndex () const

  Get the IAEA index string.
- float getMaxEnergy (ParticleType particleType) const

  Get the maximum energy for particles of a specific type.
- float getMaxWeight (ParticleType particleType) const

  Get the maximum weight for particles of a specific type.
- float getMaxX () const

Get the maximum X coordinate across all particles.
- float getMaxY () const

  Get the maximum Y coordinate across all particles.
- float getMaxZ () const

  Get the maximum Z coordinate across all particles.
- float getMeanEnergy (ParticleType particleType) const

  Get the mean energy for particles of a specific type.
- float getMeanWeight (ParticleType particleType) const

  Get the mean weight for particles of a specific type.
- float getMinEnergy (ParticleType particleType) const

  Get the minimum energy for particles of a specific type.
- float getMinWeight (ParticleType particleType) const

  Get the minimum weight for particles of a specific type.
- float getMinX () const

  Get the minimum X coordinate across all particles.
- float getMinY () const

  Get the minimum Y coordinate across all particles.
- float getMinZ () const

  Get the minimum Z coordinate across all particles.
- unsigned int getNumberOfExtraFloats () const

  Get the number of extra float values per record.
- unsigned int getNumberOfExtraLongs () const

  Get the number of extra integer values per record.
- std::uint64_t getNumberOfParticles () const

  Get the total number of particles in the phase space.
- std::uint64_t getNumberOfParticles (ParticleType particleType) const

  Get the number of particles of a specific type.
- std::uint64_t getOriginalHistories () const

  Get the number of original simulation histories.
- std::size_t getRecordLength () const

  Get the length of each particle record in bytes.
- const std::string getSection (const std::string &sectionName) const

  Get a header section value by name.
- const std::string getSection (SECTION section) const

  Get a header section value by enum.
- const std::string & getTitle () const

  Get the phase space file title.
- float getTotalWeight (ParticleType particleType) const

  Get the total weight for particles of a specific type.
- bool hasExtraFloat (EXTRA_FLOAT_TYPE type) const

  Check if a specific extra float type is included.

- bool hasExtraLong (EXTRA_LONG_TYPE type) const

  Check if a specific extra integer type is included.
- void setChecksum (std::uint64_t checksum)

  Set the data integrity checksum.
- void setConstantU (float u)

  Set the constant U direction cosine value.
- void setConstantV (float v)

  Set the constant V direction cosine value.
- void setConstantW (float w)

  Set the constant W direction cosine value.
- void setConstantWeight (float weight)

  Set the constant particle weight value.
- void setConstantX (float x)

  Set the constant X coordinate value.
- void setConstantY (float y)

  Set the constant Y coordinate value.
- void setConstantZ (float z)

  Set the constant Z coordinate value.
- void setFilePath (const std::string &filePath)

  Set the file path for the header.
- void setFileType (FileType fileType)

  Set the file type classification.
- void setIAEAIndex (const std::string &index)

  Set the IAEA index string.
- void setMaxEnergy (ParticleType particleType, float maxEnergy)

  Set the maximum energy for particles of a specific type.
- void setMaxWeight (ParticleType particleType, float maxWeight)

  Set the maximum weight for particles of a specific type.
- void setMaxX (float maxX)

  Set the maximum X coordinate boundary.
- void setMaxY (float maxY)

  Set the maximum Y coordinate boundary.
- void setMaxZ (float maxZ)

  Set the maximum Z coordinate boundary.
- void setMeanEnergy (ParticleType particleType, float meanEnergy)

  Set the mean energy for particles of a specific type.
- void setMinEnergy (ParticleType particleType, float minEnergy)

  Set the minimum energy for particles of a specific type.
- void setMinWeight (ParticleType particleType, float minWeight)

  Set the minimum weight for particles of a specific type.
- void setMinX (float minX)

Set the minimum X coordinate boundary.
- void setMinY (float minY)

    Set the minimum Y coordinate boundary.
- void setMinZ (float minZ)

    Set the minimum Z coordinate boundary.
- void setNumberOfParticles (ParticleType particleType, std::uint64_t numberOfParticles)

    Set the number of particles for a specific type.
- void setNumberOfParticles (std::uint64_t numberOfParticles)

    Set the total number of particles.
- void setOriginalHistories (std::uint64_t originalHistories)

    Set the number of original simulation histories.
- void setRecordLength (std::size_t length)

    Set the particle record length in bytes.
- void setSection (const std::string &sectionName, const std::string &sectionValue)

    Set a header section value by name.
- void setSection (SECTION section, const std::string &sectionValue)

    Set a header section value using the explicit enum type.
- void setTitle (const std::string &title)

    Set the phase space file title.
- void setTotalWeight (ParticleType particleType, float totalWeight)

    Set the total weight for particles of a specific type.
- bool uIsStored () const

    Check if U direction cosines are stored in records.
- bool vIsStored () const

    Check if V direction cosines are stored in records.
- bool weightIsStored () const

    Check if particle weights are stored in records.
- bool wIsStored () const

    Check if W direction cosines are stored in records.
- void writeHeader ()

    Write header information to file.
- bool xIsStored () const

    Check if X coordinates are stored in records.
- bool yIsStored () const

    Check if Y coordinates are stored in records.
- bool zIsStored () const

    Check if Z coordinates are stored in records.

Static Public Member Functions

- static const std::string DeterminePathToHeaderFile (const std::string &filename)

  Determine the header file path from a data file name.
- static constexpr FloatPropertyType translateExtraFloatType (IAEAHeader::EXTRA_FLOAT_TYPE type)

  Convert IAEA extra float type to ParticleZoo property type.
- static constexpr IntPropertyType translateExtraLongType (IAEAHeader::EXTRA_LONG_TYPE type)

  Convert IAEA extra 'long' type to ParticleZoo integer property type.

6.6.1 Detailed Description

Header manager for IAEA phase space files.

This class handles reading, writing, and manipulating the header information for IAEA format phase space files. It manages file metadata, particle statistics, data layout specifications, and validation checksums.

6.6.2 Member Enumeration Documentation

6.6.2.1 EXTRA_FLOAT_TYPE

```
enum class ParticleZoo::IAEAphspFile::IAEAHeader::EXTRA_FLOAT_TYPE  [strong]
```

Extra float data types for IAEA format.

Defines the types of additional floating-point data that can be stored with each particle record beyond the standard IAEA format.

Enumerator

USER_DEFINED_GENERIC_TYPE	Generic user-defined float.
XLAST	Last X position.
YLAST	Last Y position.
ZLAST	Last Z position.

6.6.2.2 EXTRA_LONG_TYPE

```
enum class ParticleZoo::IAEAphspFile::IAEAHeader::EXTRA_LONG_TYPE  [strong]
```

Extra integer data types for IAEA format.

Defines the types of additional integer data that can be stored with each particle record beyond the standard IAEA format. Refered to as "long" in the original IAEA documentation, however it is always a 32-bit integer on both 32-bit and 64-bit systems.

Enumerator

USER_DEFINED_GENERIC_TYPE	Generic user-defined integer.
INCREMENTAL_HISTORY_NUMBER	Sequential history number for tracking, tracks the number of new histories since the last particle was recorded.
EGS_LATCH	EGS-specific latch variable (see BEAMnrc User Manual for details)
PENELOPE_ILB5	PENELOPE ILB array value 1, corresponds to the generation of the particle (1 for primary, 2 for secondary, etc.)
PENELOPE_ILB4	PENELOPE ILB array value 2, corresponds to the particle type of the particle's parent (applies only if ILB1 $>$ 1)
PENELOPE_ILB3	PENELOPE ILB array value 3, corresponds to the interaction type that created the particle (applies only if ILB1 $>$ 1)
PENELOPE_ILB2	PENELOPE ILB array value 4, is non-zero if the particle is created by atomic relaxation and corresponds to the atomic transistion that created the particle.
PENELOPE_ILB1	PENELOPE ILB array value 5, a user-defined value which is passed on to all descendant particles created by this particle.

6.6.2.3 FileType

`enum class ParticleZoo::IAEAphspFile::IAEAHeader::FileType [strong]`

File type classification for IAEA phase space files.

Enumerator

PHSP_FILE	Standard phase space file.
PHSP_GENERATOR	Phase space generator file (as far as I know this is not used anywhere, but it exists in the original implementation)

6.6.2.4 SECTION

`enum class ParticleZoo::IAEAphspFile::IAEAHeader::SECTION [strong]`

Header section identifiers for IAEA format.

Defines all standard sections that can appear in an IAEA header file, used for parsing and generating header content. Includes a CUSTOM_SECTION for user-defined entries.

Enumerator

IAEA_INDEX	IAEA index code.
TITLE	File title/description.
FILE_TYPE	Either PHSP_FILE or PHSP_GENERATOR.
CHECKSUM	Data integrity checksum.
RECORD_CONTENTS	Description of record structure.
RECORD_CONSTANT	Constant values in records.
RECORD_LENGTH	Length of each particle record.
BYTE_ORDER	Byte ordering specification (endianness)
ORIGINAL_HISTORIES	Number of original simulation histories.
PARTICLES	Total particle count.
PHOTONS	Photon count and statistics.
ELECTRONS	Electron count and statistics.
POSITRONS	Positron count and statistics.
NEUTRONS	Neutron count and statistics.
PROTONS	Proton count and statistics.
TRANSPORT_PARAMETERS	Monte Carlo transport settings.
MACHINE_TYPE	Linear accelerator type.
MONTE_CARLO_CODE_VERSION	Monte Carlo code version information.
GLOBAL_PHOTON_ENERGY_CUTOFF	Global photon cutoff energy.
GLOBAL_PARTICLE_ENERGY_CUTOFF	Global particle cutoff energy.
COORDINATE_SYSTEM_DESCRIPTION	Coordinate system definition.
BEAM_NAME	Treatment beam name.
FIELD_SIZE	Radiation field dimensions.
NOMINAL_SSD	Source-to-surface distance.
MC_INPUT_FILENAME	Monte Carlo input file name.
VARIANCE_REDUCTION_TECHNIQUES	Variance reduction methods used.
INITIAL_SOURCE_DESCRIPTION	Primary source description.
PUBLISHED_REFERENCE	Publication reference.
AUTHORS	File authors.
INSTITUTION	Institution name.
LINK_VALIDATION	Validation link information.
ADDITIONAL_NOTES	Additional notes.
STATISTICAL_INFORMATION_PARTICLES	Particle statistics summary.
STATISTICAL_INFORMATION_GEOMETRY	Geometric statistics summary.
CUSTOM_SECTION	User-defined section.

6.6.3 Constructor & Destructor Documentation

6.6.3.1 IAEAHeader() [1/2]

```
ParticleZoo::IAEAphspFile::IAEAHeader::IAEAHeader (
             const std::string & filePath,
             bool newFile = false )
```

Construct header from existing IAEA header file.

Parameters

filePath	Path to the IAEA header file (.IAEAheader)
newFile	If true, creates a new header; if false, reads existing file

Exceptions

std::runtime_error	if file cannot be read or is invalid

6.6.3.2 IAEAHeader() [2/2]

```
ParticleZoo::IAEAphspFile::IAEAHeader::IAEAHeader (
             const IAEAHeader & other,
             const std::string & newFilePath )
```

Copy constructor with new file path.

Creates a new header based on an existing one but with a different file path. Resets particle counts and statistics to zero.

Parameters

other	Source header to copy from
newFilePath	Path for the new header file

6.6.4 Member Function Documentation

6.6.4.1 addExtraFloat()

```
void ParticleZoo::IAEAphspFile::IAEAHeader::addExtraFloat (
             EXTRA_FLOAT_TYPE type )  [inline]
```

Add an extra float data type to the record format.

Parameters

type	Type of additional floating-point data to include

6.6.4.2 addExtraLong()

```
void ParticleZoo::IAEAphspFile::IAEAHeader::addExtraLong (
             EXTRA_LONG_TYPE type )  [inline]
```

Add an extra integer data type to the record format.

Parameters

type	Type of additional integer data to include

6.6.4.3 checksumIsValid()

```
bool ParticleZoo::IAEAphspFile::IAEAHeader::checksumIsValid ( ) const
```

Validate the data integrity checksum This check is strict. Not only does it verify that the checksum matches the file size, but it also checks that it equals the record length multiplied by the number of particles.

Returns

true if checksum matches expected value based on file size and record length

6.6.4.4 countParticleStats()

```
void ParticleZoo::IAEAphspFile::IAEAHeader::countParticleStats (
             const Particle & particle )  [inline]
```

Update particle statistics with a new particle.

Parameters

particle	Particle to include in statistics calculations

6.6.4.5 DeterminePathToHeaderFile()

```
const std::string ParticleZoo::IAEAphspFile::IAEAHeader::DeterminePathToHeaderFile (
             const std::string & filename )  [static]
```

Determine the header file path from a data file name.

Parameters

filename	Path to the data file (.IAEAphsp)

Returns

Path to the corresponding header file (.IAEAheader)

6.6.4.6 getByteOrder()

```
ByteOrder ParticleZoo::IAEAphspFile::IAEAHeader::getByteOrder ( ) const  [inline]
```

Get the byte order for binary data (endianness)

Returns

ByteOrder specification for data interpretation

6.6.4.7 getChecksum()

```
std::uint64_t ParticleZoo::IAEAphspFile::IAEAHeader::getChecksum ( ) const  [inline]
```

Get the data integrity checksum.

Returns

Checksum value for data validation

6.6.4.8 getConstantU()

```
float ParticleZoo::IAEAphspFile::IAEAHeader::getConstantU ( ) const  [inline]
```

Get the constant U direction cosine value (when not stored per particle)

Returns

U direction cosine when not stored per particle

6.6.4.9 getConstantV()

```
float ParticleZoo::IAEAphspFile::IAEAHeader::getConstantV ( ) const  [inline]
```

Get the constant V direction cosine value (when not stored per particle)

Returns

V direction cosine when not stored per particle

6.6.4.10 getConstantW()

```
float ParticleZoo::IAEAphspFile::IAEAHeader::getConstantW ( ) const  [inline]
```

Get the constant W direction cosine value (when not implicitly stored per particle)

Returns

W direction cosine when not stored per particle

6.6.4.11 getConstantWeight()

```
float ParticleZoo::IAEAphspFile::IAEAHeader::getConstantWeight ( ) const  [inline]
```

Get the constant particle weight value (when not stored per particle)

Returns

Weight when not stored per particle

6.6.4.12 getConstantX()

```
float ParticleZoo::IAEAphspFile::IAEAHeader::getConstantX ( ) const  [inline]
```

Get the constant X coordinate value (when not stored per particle)

Returns

X coordinate when not stored per particle

6.6.4.13 getConstantY()

```
float ParticleZoo::IAEAphspFile::IAEAHeader::getConstantY ( ) const  [inline]
```

Get the constant Y coordinate value (when not stored per particle)

Returns

Y coordinate when not stored per particle

6.6.4.14 getConstantZ()

```
float ParticleZoo::IAEAphspFile::IAEAHeader::getConstantZ ( ) const  [inline]
```

Get the constant Z coordinate value (when not stored per particle)

Returns

Z coordinate when not stored per particle

6.6.4.15 getDataFilePath()

```
std::string ParticleZoo::IAEAphspFile::IAEAHeader::getDataFilePath ( ) const
```

Get the path to the associated data file.

Returns

Path to the .IAEAphsp data file

6.6.4.16 getExtraFloatType()

```
IAEAHeader::EXTRA_FLOAT_TYPE ParticleZoo::IAEAphspFile::IAEAHeader::getExtraFloatType (
             unsigned int index ) const  [inline]
```

Get the type of the extra float value at the specified index.

Parameters

index	Index of the extra float (0-based)

Returns

EXTRA_FLOAT_TYPE describing the data type

Exceptions

std::out_of_range	if index is invalid

6.6.4.17 getExtraLongType()

```
IAEAHeader::EXTRA_LONG_TYPE ParticleZoo::IAEAphspFile::IAEAHeader::getExtraLongType (
             unsigned int index ) const  [inline]
```

Get the type of the extra integer value at the specified index.

Parameters

index	Index of the extra integer (0-based)

Returns

EXTRA_LONG_TYPE describing the data type

Exceptions

std::out_of_range	if index is invalid

6.6.4.18 getFileType()

```
IAEAHeader::FileType ParticleZoo::IAEAphspFile::IAEAHeader::getFileType ( ) const  [inline]
```

Get the file type classification.

Returns

FileType indicating PHSP_FILE or PHSP_GENERATOR

6.6.4.19 getHeaderFilePath()

```
std::string ParticleZoo::IAEAphspFile::IAEAHeader::getHeaderFilePath ( ) const  [inline]
```

Get the path to the header file.

Returns

Path to the .IAEAheader file

6.6.4.20 getIAEAIndex()

```
std::string ParticleZoo::IAEAphspFile::IAEAHeader::getIAEAIndex ( ) const  [inline]
```

Get the IAEA index string.

Returns

IAEA index (preserved with leading zeros if present)

6.6.4.21 getMaxEnergy()

```
float ParticleZoo::IAEAphspFile::IAEAHeader::getMaxEnergy (
            ParticleType particleType ) const
```

Get the maximum energy for particles of a specific type.

Parameters

particleType	Type of particle to query

Returns

Maximum kinetic energy for the particle type

6.6.4.22 getMaxWeight()

```
float ParticleZoo::IAEAphspFile::IAEAHeader::getMaxWeight (
            ParticleType particleType ) const
```

Get the maximum weight for particles of a specific type.

Parameters

particleType	Type of particle to query

Returns

Maximum weight value for the particle type

6.6.4.23 getMaxX()

```
float ParticleZoo::IAEAphspFile::IAEAHeader::getMaxX ( ) const  [inline]
```

Get the maximum X coordinate across all particles.

Returns

Maximum X value in the phase space

6.6.4.24 getMaxY()

```
float ParticleZoo::IAEAphspFile::IAEAHeader::getMaxY ( ) const  [inline]
```

Get the maximum Y coordinate across all particles.

Returns

Maximum Y value in the phase space

6.6.4.25 getMaxZ()

```
float ParticleZoo::IAEAphspFile::IAEAHeader::getMaxZ ( ) const  [inline]
```

Get the maximum Z coordinate across all particles.

Returns

Maximum Z value in the phase space

6.6.4.26 getMeanEnergy()

```
float ParticleZoo::IAEAphspFile::IAEAHeader::getMeanEnergy (
            ParticleType particleType ) const
```

Get the mean energy for particles of a specific type.

Parameters

particleType	Type of particle to query

Returns

Average kinetic energy for the particle type

6.6.4.27 getMeanWeight()

```
float ParticleZoo::IAEAphspFile::IAEAHeader::getMeanWeight (
            ParticleType particleType ) const
```

Get the mean weight for particles of a specific type.

Parameters

particleType	Type of particle to query

Returns

Average weight value for the particle type

6.6.4.28 getMinEnergy()

```
float ParticleZoo::IAEAphspFile::IAEAHeader::getMinEnergy (
            ParticleType particleType ) const
```

Get the minimum energy for particles of a specific type.

Parameters

particleType	Type of particle to query

Returns

Minimum kinetic energy for the particle type

6.6.4.29 getMinWeight()

```
float ParticleZoo::IAEAphspFile::IAEAHeader::getMinWeight (
            ParticleType particleType ) const
```

Get the minimum weight for particles of a specific type.

Parameters

particleType	Type of particle to query

Returns

Minimum weight value for the particle type

6.6.4.30 getMinX()

```
float ParticleZoo::IAEAphspFile::IAEAHeader::getMinX ( ) const  [inline]
```

Get the minimum X coordinate across all particles.

Returns

Minimum X value in the phase space

6.6.4.31 getMinY()

```
float ParticleZoo::IAEAphspFile::IAEAHeader::getMinY ( ) const  [inline]
```

Get the minimum Y coordinate across all particles.

Returns

Minimum Y value in the phase space

6.6.4.32 getMinZ()

```
float ParticleZoo::IAEAphspFile::IAEAHeader::getMinZ ( ) const  [inline]
```

Get the minimum Z coordinate across all particles.

Returns

Minimum Z value in the phase space

6.6.4.33 getNumberOfExtraFloats()

```
unsigned int ParticleZoo::IAEAphspFile::IAEAHeader::getNumberOfExtraFloats ( ) const  [inline]
```

Get the number of extra float values per record.

Returns

Count of additional floating-point values

6.6.4.34 getNumberOfExtraLongs()

```
unsigned int ParticleZoo::IAEAphspFile::IAEAHeader::getNumberOfExtraLongs ( ) const  [inline]
```

Get the number of extra integer values per record.

Returns

Count of additional integer values

6.6.4.35 getNumberOfParticles() [1/2]

```
std::uint64_t ParticleZoo::IAEAphspFile::IAEAHeader::getNumberOfParticles ( ) const  [inline]
```

Get the total number of particles in the phase space.

Returns

Total particle count across all types

6.6.4.36 getNumberOfParticles() [2/2]

```
std::uint64_t ParticleZoo::IAEAphspFile::IAEAHeader::getNumberOfParticles (
             ParticleType particleType ) const
```

Get the number of particles of a specific type.

Parameters

particleType	Type of particle to count

Returns

Number of particles of the specified type

6.6.4.37 getOriginalHistories()

```
std::uint64_t ParticleZoo::IAEAphspFile::IAEAHeader::getOriginalHistories ( ) const  [inline]
```

Get the number of original simulation histories.

Returns

The number of primary histories used to generate the phase space

6.6.4.38 getRecordLength()

```
std::size_t ParticleZoo::IAEAphspFile::IAEAHeader::getRecordLength ( ) const  [inline]
```

Get the length of each particle record in bytes.

Returns

Record length in bytes

6.6.4.39 getSection() [1/2]

```
const std::string ParticleZoo::IAEAphspFile::IAEAHeader::getSection (
            const std::string & sectionName ) const
```

Get a header section value by name.

Parameters

sectionName	Name of the section to retrieve

Returns

Section content as string, "UNKNOWN" if not found

6.6.4.40 getSection() [2/2]

```
const std::string ParticleZoo::IAEAphspFile::IAEAHeader::getSection (
            SECTION section ) const
```

Get a header section value by enum.

Parameters

section	Section identifier to retrieve

Returns

Section content as string, empty if not found

6.6.4.41 getTitle()

```
const std::string & ParticleZoo::IAEAphspFile::IAEAHeader::getTitle ( ) const  [inline]
```

Get the phase space file title.

Returns

Title string describing the phase space file

6.6.4.42 getTotalWeight()

```
float ParticleZoo::IAEAphspFile::IAEAHeader::getTotalWeight (
            ParticleType particleType ) const
```

Get the total weight for particles of a specific type.

Parameters

particleType	Type of particle to query

Returns

Sum of all weights for the particle type

6.6.4.43 hasExtraFloat()

```
bool ParticleZoo::IAEAphspFile::IAEAHeader::hasExtraFloat (
            EXTRA_FLOAT_TYPE type ) const  [inline]
```

Check if a specific extra float type is included.

Parameters

type	Extra float type to check for

Returns

true if the type is included in the record format

6.6.4.44 hasExtraLong()

```
bool ParticleZoo::IAEAphspFile::IAEAHeader::hasExtraLong (
             EXTRA_LONG_TYPE type ) const  [inline]
```

Check if a specific extra integer type is included.

Parameters

type	Extra integer type to check for

Returns

true if the type is included in the record format

6.6.4.45 setChecksum()

```
void ParticleZoo::IAEAphspFile::IAEAHeader::setChecksum (
             std::uint64_t checksum )  [inline]
```

Set the data integrity checksum.

Parameters

checksum	New checksum value

6.6.4.46 setConstantU()

```
void ParticleZoo::IAEAphspFile::IAEAHeader::setConstantU (
             float u )  [inline]
```

Set the constant U direction cosine value.

Parameters

u	U direction cosine for all particles

6.6.4.47 setConstantV()

```
void ParticleZoo::IAEAphspFile::IAEAHeader::setConstantV (
            float v )  [inline]
```

Set the constant V direction cosine value.

Parameters

v	V direction cosine for all particles

6.6.4.48 setConstantW()

```
void ParticleZoo::IAEAphspFile::IAEAHeader::setConstantW (
            float w )  [inline]
```

Set the constant W direction cosine value.

Parameters

w	W direction cosine for all particles

6.6.4.49 setConstantWeight()

```
void ParticleZoo::IAEAphspFile::IAEAHeader::setConstantWeight (
            float weight )  [inline]
```

Set the constant particle weight value.

Parameters

weight	Weight for all particles

6.6.4.50 setConstantX()

```
void ParticleZoo::IAEAphspFile::IAEAHeader::setConstantX (
             float x )  [inline]
```

Set the constant X coordinate value.

Parameters

x	X coordinate for all particles

6.6.4.51 setConstantY()

```
void ParticleZoo::IAEAphspFile::IAEAHeader::setConstantY (
             float y )  [inline]
```

Set the constant Y coordinate value.

Parameters

y	Y coordinate for all particles

6.6.4.52 setConstantZ()

```
void ParticleZoo::IAEAphspFile::IAEAHeader::setConstantZ (
             float z )  [inline]
```

Set the constant Z coordinate value.

Parameters

z	Z coordinate for all particles

6.6.4.53 setFilePath()

```
void ParticleZoo::IAEAphspFile::IAEAHeader::setFilePath (
             const std::string & filePath )  [inline]
```

Set the file path for the header.

Parameters

filePath	New path to the .IAEAheader file

6.6.4.54 setFileType()

```
void ParticleZoo::IAEAphspFile::IAEAHeader::setFileType (
             IAEAHeader::FileType fileType )  [inline]
```

Set the file type classification.

Parameters

fileType	Type specification (PHSP_FILE or PHSP_GENERATOR)

6.6.4.55 setIAEAIndex()

```
void ParticleZoo::IAEAphspFile::IAEAHeader::setIAEAIndex (
             const std::string & index )  [inline]
```

Set the IAEA index string.

Parameters

index	New IAEA index identifier

6.6.4.56 setMaxEnergy()

```
void ParticleZoo::IAEAphspFile::IAEAHeader::setMaxEnergy (
             ParticleType particleType,
             float maxEnergy )  [inline]
```

Set the maximum energy for particles of a specific type.

Parameters

particleType	Type of particle to set statistics for
maxEnergy	Maximum kinetic energy for this particle type

6.6.4.57 setMaxWeight()

```
void ParticleZoo::IAEAphspFile::IAEAHeader::setMaxWeight (
            ParticleType particleType,
            float maxWeight )  [inline]
```

Set the maximum weight for particles of a specific type.

Parameters

particleType	Type of particle to set statistics for
maxWeight	Maximum weight value for this particle type

6.6.4.58 setMaxX()

```
void ParticleZoo::IAEAphspFile::IAEAHeader::setMaxX (
            float maxX )  [inline]
```

Set the maximum X coordinate boundary.

Parameters

maxX	Maximum X value in the phase space

6.6.4.59 setMaxY()

```
void ParticleZoo::IAEAphspFile::IAEAHeader::setMaxY (
            float maxY )  [inline]
```

Set the maximum Y coordinate boundary.

Parameters

maxY	Maximum Y value in the phase space

6.6.4.60 setMaxZ()

```
void ParticleZoo::IAEAphspFile::IAEAHeader::setMaxZ (
             float maxZ )  [inline]
```

Set the maximum Z coordinate boundary.

Parameters

maxZ	Maximum Z value in the phase space

6.6.4.61 setMeanEnergy()

```
void ParticleZoo::IAEAphspFile::IAEAHeader::setMeanEnergy (
             ParticleType particleType,
             float meanEnergy )  [inline]
```

Set the mean energy for particles of a specific type.

Parameters

particleType	Type of particle to set statistics for
meanEnergy	Average kinetic energy for this particle type

6.6.4.62 setMinEnergy()

```
void ParticleZoo::IAEAphspFile::IAEAHeader::setMinEnergy (
             ParticleType particleType,
             float minEnergy )  [inline]
```

Set the minimum energy for particles of a specific type.

Parameters

particleType	Type of particle to set statistics for
minEnergy	Minimum kinetic energy for this particle type

6.6.4.63 setMinWeight()

```
void ParticleZoo::IAEAphspFile::IAEAHeader::setMinWeight (
             ParticleType particleType,
             float minWeight )  [inline]
```

Set the minimum weight for particles of a specific type.

Parameters

particleType	Type of particle to set statistics for
minWeight	Minimum weight value for this particle type

6.6.4.64 setMinX()

```
void ParticleZoo::IAEAphspFile::IAEAHeader::setMinX (
             float minX )  [inline]
```

Set the minimum X coordinate boundary.

Parameters

minX	Minimum X value in the phase space

6.6.4.65 setMinY()

```
void ParticleZoo::IAEAphspFile::IAEAHeader::setMinY (
             float minY )  [inline]
```

Set the minimum Y coordinate boundary.

Parameters

minY	Minimum Y value in the phase space

6.6.4.66 setMinZ()

```
void ParticleZoo::IAEAphspFile::IAEAHeader::setMinZ (
            float minZ )  [inline]
```

Set the minimum Z coordinate boundary.

Parameters

minZ	Minimum Z value in the phase space

6.6.4.67 setNumberOfParticles() [1/2]

```
void ParticleZoo::IAEAphspFile::IAEAHeader::setNumberOfParticles (
            ParticleType particleType,
            std::uint64_t numberOfParticles )  [inline]
```

Set the number of particles for a specific type.

Parameters

particleType	Type of particle to set count for
numberOfParticles	Number of particles of this type

6.6.4.68 setNumberOfParticles() [2/2]

```
void ParticleZoo::IAEAphspFile::IAEAHeader::setNumberOfParticles (
            std::uint64_t numberOfParticles )  [inline]
```

Set the total number of particles.

Parameters

numberOfParticles	Total particle count across all types

6.6.4.69 setOriginalHistories()

```
void ParticleZoo::IAEAphspFile::IAEAHeader::setOriginalHistories (
             std::uint64_t originalHistories )  [inline]
```

Set the number of original simulation histories.

Parameters

originalHistories	Count of primary histories

6.6.4.70 setRecordLength()

```
void ParticleZoo::IAEAphspFile::IAEAHeader::setRecordLength (
             std::size_t length )  [inline]
```

Set the particle record length in bytes.

Parameters

length	New record length for each particle

6.6.4.71 setSection() `[1/2]`

```
void ParticleZoo::IAEAphspFile::IAEAHeader::setSection (
             const std::string & sectionName,
             const std::string & sectionValue )
```

Set a header section value by name.

Parameters

sectionName	Name of the section to set
sectionValue	Content to store in the section

6.6.4.72 setSection() [2/2]

```
void ParticleZoo::IAEAphspFile::IAEAHeader::setSection (
            SECTION section,
            const std::string & sectionValue )
```

Set a header section value using the explicit enum type.

Parameters

section	Section identifier to set
sectionValue	Content to store in the section

6.6.4.73 setTitle()

```
void ParticleZoo::IAEAphspFile::IAEAHeader::setTitle (
            const std::string & title ) [inline]
```

Set the phase space file title.

Parameters

title	New title for the file

6.6.4.74 setTotalWeight()

```
void ParticleZoo::IAEAphspFile::IAEAHeader::setTotalWeight (
            ParticleType particleType,
            float totalWeight ) [inline]
```

Set the total weight for particles of a specific type.

Parameters

particleType	Type of particle to set statistics for
totalWeight	Sum of all weights for this particle type

6.6.4.75 translateExtraFloatType()

```
constexpr FloatPropertyType ParticleZoo::IAEAphspFile::IAEAHeader::translateExtraFloatType (
            IAEAHeader::EXTRA_FLOAT_TYPE type )  [static], [constexpr]
```

Convert IAEA extra float type to ParticleZoo property type.

Parameters

type	IAEA-specific extra float type

Returns

Corresponding ParticleZoo FloatPropertyType

6.6.4.76 translateExtraLongType()

```
constexpr IntPropertyType ParticleZoo::IAEAphspFile::IAEAHeader::translateExtraLongType (
            IAEAHeader::EXTRA_LONG_TYPE type )  [static], [constexpr]
```

Convert IAEA extra 'long' type to ParticleZoo integer property type.

Parameters

type	IAEA-specific extra 'long' type

Returns

Corresponding ParticleZoo IntPropertyType

6.6.4.77 uIsStored()

```
bool ParticleZoo::IAEAphspFile::IAEAHeader::uIsStored ( ) const  [inline]
```

Check if U direction cosines are stored in records.

Returns

true if U values are stored, false if constant

6.6.4.78 vIsStored()

```
bool ParticleZoo::IAEAphspFile::IAEAHeader::vIsStored ( ) const  [inline]
```

Check if V direction cosines are stored in records.

Returns

true if V values are stored, false if constant

6.6.4.79 weightIsStored()

```
bool ParticleZoo::IAEAphspFile::IAEAHeader::weightIsStored ( ) const  [inline]
```

Check if particle weights are stored in records.

Returns

true if weights are stored, false if constant

6.6.4.80 wIsStored()

```
bool ParticleZoo::IAEAphspFile::IAEAHeader::wIsStored ( ) const  [inline]
```

Check if W direction cosines are stored in records.

Note

W being 'stored' means that it is not a constant value recorded in the header. The 'stored' value is only implicitly stored and is actually calculated as needed from U and V.

Returns

true if W values are stored, false if constant

6.6.4.81 writeHeader()

```
void ParticleZoo::IAEAphspFile::IAEAHeader::writeHeader ( )
```

Write header information to file.

Writes the complete header information to the associated .IAEAheader file, including all sections, particle statistics, and metadata.

Exceptions

std::runtime_error	if file cannot be written

6.6.4.82 xIsStored()

```
bool ParticleZoo::IAEAphspFile::IAEAHeader::xIsStored ( ) const  [inline]
```

Check if X coordinates are stored in records.

Returns

true if X values are stored, false if constant

6.6.4.83 yIsStored()

```
bool ParticleZoo::IAEAphspFile::IAEAHeader::yIsStored ( ) const  [inline]
```

Check if Y coordinates are stored in records.

Returns

true if Y values are stored, false if constant

6.6.4.84 zIsStored()

```
bool ParticleZoo::IAEAphspFile::IAEAHeader::zIsStored ( ) const  [inline]
```

Check if Z coordinates are stored in records.

Returns

true if Z values are stored, false if constant

The documentation for this class was generated from the following files:

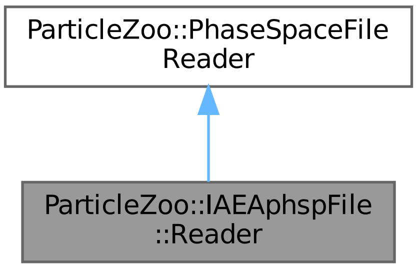
- include/particlezoo/IAEA/IAEAHeader.h
- src/IAEA/IAEAHeader.cc

6.7 ParticleZoo::IAEAphspFile::Reader Class Reference

Reader for IAEA format phase space files.

```
#include <particlezoo/IAEA/IAEAphspFile.h>
```

Inheritance diagram for ParticleZoo::IAEAphspFile::Reader:


ParticleZoo::PhaseSpaceFile Reader
ParticleZoo::IAEAphspFile ::Reader

Public Member Functions

- Reader (const std::string &filename, const UserOptions &options=UserOptions{})

  Construct reader for IAEA phase space file.
- const IAEAHeader & getHeader () const

  Get access to the IAEA header information.
- std::uint64_t getNumberOfOriginalHistories () const override

  Get the number of original simulation histories.
- std::uint64_t getNumberOfParticles () const override

  Get the total number of particles in the phase space.
- std::uint64_t getNumberOfParticles (ParticleType particleType) const

  Get the number of particles of a specific type.

Public Member Functions inherited from ParticleZoo::PhaseSpaceFileReader

- PhaseSpaceFileReader (const std::string &phspFormat, const std::string &fileName, const UserOptions &user↩ Options, FormatType formatType=FormatType::BINARY, const FixedValues fixedValues=FixedValues(), unsigned int bufferSize=DEFAULT_BUFFER_SIZE)

  Construct a new Phase Space File Reader object.
- virtual ∼PhaseSpaceFileReader ()

  Destroy the Phase Space File Reader object.
- void close ()

  Close the phase space file and clean up resources.
- float getConstantPx () const

  Get the constant X-component of the direction unit vector (if constant).
- float getConstantPy () const

  Get the constant Y-component of the direction unit vector (if constant).
- float getConstantPz () const

  Get the constant Z-component of the direction unit vector (if constant).
- float getConstantWeight () const

  Get the constant statistical weight value (if constant).
- float getConstantX () const

  Get the constant X coordinate value (if constant).
- float getConstantY () const

  Get the constant Y coordinate value (if constant).
- float getConstantZ () const

  Get the constant Z coordinate value (if constant).
- const std::string getFileName () const

  Get the filename of the phase space file being read.
- std::uint64_t getFileSize () const

  Get the size of the phase space file in bytes.
- const FixedValues getFixedValues () const

  Get the fixed values configuration.
- virtual std::uint64_t getHistoriesRead ()

  Get the number of Monte Carlo histories that have been read so far. If the end of the file has been reached, this will return the total number of original histories unless more histories than expected have already been read - in which case it returns the actual count.
- Particle getNextParticle ()

  Get the next particle from the phase space file.
- virtual std::uint64_t getParticlesRead ()

  Get the number of particles that have been read so far.
- const std::string getPHSPFormat () const

  Get the phase space file format identifier.
- virtual bool hasMoreParticles ()

  Check if there are more particles to read in the file.

- bool isPxConstant () const

  Check if the X-component of momentum is constant for all particles.
- bool isPyConstant () const

  Check if the Y-component of momentum is constant for all particles.
- bool isPzConstant () const

  Check if the Z-component of momentum is constant for all particles.
- bool isWeightConstant () const

  Check if the statistical weight is constant for all particles.
- bool isXConstant () const

  Check if the X coordinate is constant for all particles.
- bool isYConstant () const

  Check if the Y coordinate is constant for all particles.
- bool isZConstant () const

  Check if the Z coordinate is constant for all particles.
- void moveToParticle (std::uint64_t particleIndex)

  Move the file position to a specific particle index.
- void setCommentMarkers (const std::vector< std::string > &commentMarkers)

  Set comment markers for ASCII format files.

Static Public Member Functions

- static std::vector< CLICommand > getFormatSpecificCLICommands ()

  Get format-specific command-line options.

Static Public Member Functions inherited from ParticleZoo::PhaseSpaceFileReader

- static std::vector< CLICommand > getCLICommands ()

  Get command line interface commands supported by this reader.

Protected Member Functions

- std::size_t getParticleRecordLength () const override

  Get the length of each particle record in bytes.
- std::size_t getParticleRecordStartOffset () const override

  Get the byte offset where particle records start.
- Particle readBinaryParticle (ByteBuffer &buffer) override

  Read and decode a single particle from binary data.

Protected Member Functions inherited from ParticleZoo::PhaseSpaceFileReader

- double calcThirdUnitComponent (double &u, double &v) const

  Calculate the third component of a unit vector from two components (double precision).
- float calcThirdUnitComponent (float &u, float &v) const

  Calculate the third component of a unit vector from two components (float precision).
- const ByteBuffer getHeaderData ()

  Get the file header data as a byte buffer.
- const ByteBuffer getHeaderData (std::size_t headerSize)

  Get a specific amount of header data as a byte buffer.
- virtual std::size_t getMaximumASCIILineLength () const

  Get the maximum line length for ASCII format files.
- Particle getNextParticle (bool countParticleInStatistics)

  Get the next particle with optional statistics counting control.
- std::size_t getNumberOfEntriesInFile () const

  Get the number of particle records that fit in the file.
- virtual std::uint64_t getParticlesRead (bool includeSkippedParticles)

  Get the number of particles read with optional inclusion of skipped particles (including pseudo-particles).
- const UserOptions & getUserOptions () const

  Get the user options that were passed to the constructor.
- virtual Particle readASCIIParticle (const std::string &line)

  Read a particle from ASCII data.
- virtual Particle readParticleManually ()

  Read a particle manually (for formats requiring third-party I/O).
- void setByteOrder (ByteOrder byteOrder)

  Set the byte order for binary data interpretation.
- void setConstantPx (float Px)

  Set a constant X-component of the direction unit vector for all particles.
- void setConstantPy (float Py)

  Set a constant Y-component of the direction unit vector for all particles.
- void setConstantPz (float Pz)

  Set a constant Z-component of the direction unit vector for all particles.
- void setConstantWeight (float weight)

  Set a constant statistical weight for all particles.
- void setConstantX (float X)

  Set a constant X coordinate value for all particles.
- void setConstantY (float Y)

  Set a constant Y coordinate value for all particles.
- void setConstantZ (float Z)

  Set a constant Z coordinate value for all particles.

6.7.1 Detailed Description

Reader for IAEA format phase space files.

Provides functionality to read phase space data from IAEA format files, handling both header parsing and binary particle data extraction. Supports all standard IAEA features including extra float/long data types.

6.7.2 Constructor & Destructor Documentation

6.7.2.1 Reader()

```
ParticleZoo::IAEAphspFile::Reader::Reader (
            const std::string & filename,
            const UserOptions & options = UserOptions{} )
```

Construct reader for IAEA phase space file.

Parameters

filename	Path to the IAEA phase space data file (.IAEAphsp)
options	User-specified options for reading behavior

Exceptions

std::runtime_error	if file cannot be opened or header is invalid

6.7.3 Member Function Documentation

6.7.3.1 getFormatSpecificCLICommands()

```
std::vector< CLICommand > ParticleZoo::IAEAphspFile::Reader::getFormatSpecificCLICommands ( )
[static]
```

Get format-specific command-line options.

Returns

Vector of CLI commands supported by IAEA reader

6.7.3.2 getHeader()

`const IAEAHeader & ParticleZoo::IAEAphspFile::Reader::getHeader ( ) const [inline]`

Get access to the IAEA header information.

Returns

Reference to the header containing file metadata

6.7.3.3 getNumberOfOriginalHistories()

`std::uint64_t ParticleZoo::IAEAphspFile::Reader::getNumberOfOriginalHistories ( ) const [inline], [override], [virtual]`

Get the number of original simulation histories.

Returns

Count of primary histories used in the simulation

Implements ParticleZoo::PhaseSpaceFileReader.

6.7.3.4 getNumberOfParticles() `[1/2]`

`std::uint64_t ParticleZoo::IAEAphspFile::Reader::getNumberOfParticles ( ) const [inline], [override], [virtual]`

Get the total number of particles in the phase space.

Returns

Total particle count across all types

Implements ParticleZoo::PhaseSpaceFileReader.

6.7.3.5 getNumberOfParticles() `[2/2]`

```
std::uint64_t ParticleZoo::IAEAphspFile::Reader::getNumberOfParticles (
            ParticleType particleType ) const [inline]
```

Get the number of particles of a specific type.

Parameters

particleType	Type of particle to count

Returns

Number of particles of the specified type

6.7.3.6 getParticleRecordLength()

```
std::size_t ParticleZoo::IAEAphspFile::Reader::getParticleRecordLength ( ) const  [inline], [override],
[protected], [virtual]
```

Get the length of each particle record in bytes.

Returns

Size of each particle record as defined in header

Reimplemented from ParticleZoo::PhaseSpaceFileReader.

6.7.3.7 getParticleRecordStartOffset()

```
std::size_t ParticleZoo::IAEAphspFile::Reader::getParticleRecordStartOffset ( ) const  [inline],
[override], [protected], [virtual]
```

Get the byte offset where particle records start.

Returns

Starting offset for particle data (0 for IAEA format)

Reimplemented from ParticleZoo::PhaseSpaceFileReader.

6.7.3.8 readBinaryParticle()

```
Particle ParticleZoo::IAEAphspFile::Reader::readBinaryParticle (
             ByteBuffer & buffer )  [override], [protected], [virtual]
```

Read and decode a single particle from binary data.

Parameters

buffer	Binary buffer containing particle data

Returns

Decoded Particle object with all properties

Exceptions

std::runtime_error	if particle data is corrupted or invalid

Reimplemented from ParticleZoo::PhaseSpaceFileReader.

The documentation for this class was generated from the following files:

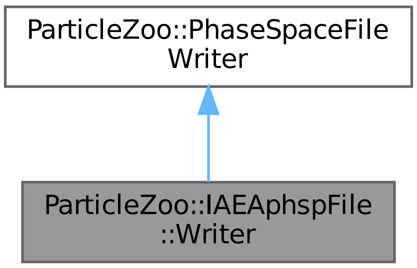
- include/particlezoo/IAEA/IAEAphspFile.h
- src/IAEA/IAEAphspFile.cc

6.8 ParticleZoo::IAEAphspFile::Writer Class Reference

Writer for IAEA format phase space files.

```
#include <particlezoo/IAEA/IAEAphspFile.h>
```

Inheritance diagram for ParticleZoo::IAEAphspFile::Writer:


ParticleZoo::PhaseSpaceFile Writer
ParticleZoo::IAEAphspFile ::Writer

Public Member Functions

- Writer (const std::string &filename, const IAEAHeader &templateHeader)

  Construct writer using an existing header as template.
- Writer (const std::string &filename, const UserOptions &userOptions=UserOptions{}, const FixedValues &fixed↩ Values=FixedValues{})

  Construct writer for new IAEA phase space file.
- IAEAHeader & getHeader ()

  Get access to the IAEA header for configuration.
- std::uint64_t getMaximumSupportedParticles () const override

  Get the maximum number of particles this format can store.
- void setNumberOfOriginalHistories (std::uint64_t numberOfHistories)

  Set the number of original simulation histories.

Public Member Functions inherited from ParticleZoo::PhaseSpaceFileWriter

- PhaseSpaceFileWriter (const std::string &phspFormat, const std::string &fileName, const UserOptions &user↩ Options, FormatType formatType=FormatType::BINARY, const FixedValues fixedValues=FixedValues(), unsigned int bufferSize=DEFAULT_BUFFER_SIZE)

  Construct a new Phase Space File Writer object.
- virtual ∼PhaseSpaceFileWriter ()

  Destroy the Phase Space File Writer object.
- void addAdditionalHistories (std::uint64_t additionalHistories)

  Add additional Monte Carlo histories to the count.
- void close ()

  Close the phase space file and finalize writing.
- ByteOrder getByteOrder () const

  Get the byte order used for binary data writing.
- float getConstantPx () const

  Get the constant X-component of the direction unit vector (if constant).
- float getConstantPy () const

  Get the constant Y-component of the direction unit vector (if constant).
- float getConstantPz () const

  Get the constant Z-component of the direction unit vector (if constant).
- float getConstantWeight () const

  Get the constant statistical weight value (if constant).
- float getConstantX () const

  Get the constant X coordinate value (if constant).
- float getConstantY () const

  Get the constant Y coordinate value (if constant).
- float getConstantZ () const

  Get the constant Z coordinate value (if constant).
- const std::string getFileName () const

  Get the filename where the phase space file is being written.
- const FixedValues getFixedValues () const

  Get the fixed values configuration.
- virtual std::uint64_t getHistoriesWritten () const

  Get the number of Monte Carlo histories that have been written.
- std::uint64_t getParticlesWritten () const

  Get the number of particles that have been written to the file.
- const std::string getPHSPFormat () const

  Get the phase space file format identifier.
- bool isPxConstant () const

  Check if the X-component of the direction unit vector is set to a constant value for all particles.
- bool isPyConstant () const

  Check if the Y-component of the direction unit vector is set to a constant value for all particles.

- bool isPzConstant () const

  Check if the Z-component of the direction unit vector is set to a constant value for all particles.
- bool isWeightConstant () const

  Check if the statistical weight is set to a constant value for all particles.
- bool isXConstant () const

  Check if the X coordinate is set to a constant value for all particles.
- bool isYConstant () const

  Check if the Y coordinate is set to a constant value for all particles.
- bool isZConstant () const

  Check if the Z coordinate is set to a constant value for all particles.
- virtual void writeParticle (Particle particle)

  Write a particle to the phase space file.

Static Public Member Functions

- static std::vector< CLICommand > getFormatSpecificCLICommands ()

  Get format-specific command-line options.

Static Public Member Functions inherited from ParticleZoo::PhaseSpaceFileWriter

- static std::vector< CLICommand > getCLICommands ()

  Get command line interface commands supported by this writer.

Protected Member Functions

- bool canHaveConstantPx () const override

  Check if constant X momentum components are supported.
- bool canHaveConstantPy () const override

  Check if constant Y momentum components are supported.
- bool canHaveConstantPz () const override

  Check if constant Z momentum components are supported.
- bool canHaveConstantWeight () const override

  Check if constant particle weights are supported.
- bool canHaveConstantX () const override

  Check if constant X coordinates are supported.
- bool canHaveConstantY () const override

  Check if constant Y coordinates are supported.
- bool canHaveConstantZ () const override

  Check if constant Z coordinates are supported.

- void fixedValuesHaveChanged () override

  Handle changes to fixed/constant values.
- std::size_t getParticleRecordLength () const override

  Get the length of each particle record in bytes.
- void writeBinaryParticle (ByteBuffer &buffer, Particle &particle) override

  Encode and write a single particle to binary data.
- void writeHeaderData (ByteBuffer &buffer) override

  Write header data to the output buffer (not used for IAEA)

Protected Member Functions inherited from ParticleZoo::PhaseSpaceFileWriter

- virtual bool accountForAdditionalHistories (std::uint64_t additionalHistories)

  Handle accounting for simulation histories that produced no particles.
- virtual bool canWritePseudoParticlesExplicitly () const

  Check if this format can write pseudo-particles explicitly.
- virtual size_t getMaximumASCIILineLength () const

  Get the maximum line length for ASCII format files.
- virtual std::size_t getParticleRecordStartOffset () const

  Get the byte offset where particle records start in the file.
- virtual std::uint64_t getPendingHistories () const

  Get the number of pending histories to account for.
- const UserOptions & getUserOptions () const

  Get the user options that were passed to the constructor.
- void setByteOrder (ByteOrder byteOrder)

  Set the byte order for binary data writing.
- void setConstantPx (float Px)

  Set a constant X-component of the direction unit vector for all particles.
- void setConstantPy (float Py)

  Set a constant Y-component of the direction unit vector for all particles.
- void setConstantPz (float Pz)

  Set a constant Z-component of the direction unit vector for all particles.
- void setConstantWeight (float weight)

  Set a constant statistical weight for all particles.
- void setConstantX (float X)

  Set a constant X coordinate value for all particles.
- void setConstantY (float Y)

  Set a constant Y coordinate value for all particles.
- void setConstantZ (float Z)

  Set a constant Z coordinate value for all particles.
- virtual const std::string writeASCIIParticle (Particle &particle)

  Write a particle in ASCII format as a string.
- virtual void writeParticleManually (Particle &particle)

  Write a particle manually (for formats requiring third-party I/O).

6.8.1 Detailed Description

Writer for IAEA format phase space files.

Provides functionality to write phase space data to IAEA format files, handling header generation and binary particle data encoding. Supports all standard IAEA features and optional data types.

6.8.2 Constructor & Destructor Documentation

6.8.2.1 Writer() [1/2]

```
ParticleZoo::IAEAphspFile::Writer::Writer (
            const std::string & filename,
            const UserOptions & userOptions = UserOptions{},
            const FixedValues & fixedValues = FixedValues{} )
```

Construct writer for new IAEA phase space file.

Parameters

filename	Path for the new IAEA phase space data file (.IAEAphsp)
userOptions	User-specified options for writing behavior
fixedValues	Constant values to optimize storage

Exceptions

std::runtime_error	if file cannot be created

6.8.2.2 Writer() [2/2]

```
ParticleZoo::IAEAphspFile::Writer::Writer (
            const std::string & filename,
            const IAEAHeader & templateHeader )
```

Construct writer using an existing header as template.

Parameters

filename	Path for the new IAEA phase space data file (.IAEAphsp)
templateHeader	Existing header to copy configuration from

6.8.3 Member Function Documentation

6.8.3.1 canHaveConstantPx()

`bool ParticleZoo::IAEAphspFile::Writer::canHaveConstantPx ( ) const [inline], [override], [protected], [virtual]`

Check if constant X momentum components are supported.

Returns

true (IAEA format supports constant U direction cosines)

Reimplemented from ParticleZoo::PhaseSpaceFileWriter.

6.8.3.2 canHaveConstantPy()

`bool ParticleZoo::IAEAphspFile::Writer::canHaveConstantPy ( ) const [inline], [override], [protected], [virtual]`

Check if constant Y momentum components are supported.

Returns

true (IAEA format supports constant V direction cosines)

Reimplemented from ParticleZoo::PhaseSpaceFileWriter.

6.8.3.3 canHaveConstantPz()

`bool ParticleZoo::IAEAphspFile::Writer::canHaveConstantPz ( ) const [inline], [override], [protected], [virtual]`

Check if constant Z momentum components are supported.

Returns

true (IAEA format supports constant W direction cosines)

Reimplemented from ParticleZoo::PhaseSpaceFileWriter.

6.8.3.4 canHaveConstantWeight()

```
bool ParticleZoo::IAEAphspFile::Writer::canHaveConstantWeight ( ) const  [inline], [override],
[protected], [virtual]
```

Check if constant particle weights are supported.

Returns

true (IAEA format supports constant weight values)

Reimplemented from ParticleZoo::PhaseSpaceFileWriter.

6.8.3.5 canHaveConstantX()

```
bool ParticleZoo::IAEAphspFile::Writer::canHaveConstantX ( ) const  [inline], [override], [protected],
[virtual]
```

Check if constant X coordinates are supported.

Returns

true (IAEA format supports constant X values)

Reimplemented from ParticleZoo::PhaseSpaceFileWriter.

6.8.3.6 canHaveConstantY()

```
bool ParticleZoo::IAEAphspFile::Writer::canHaveConstantY ( ) const  [inline], [override], [protected],
[virtual]
```

Check if constant Y coordinates are supported.

Returns

true (IAEA format supports constant Y values)

Reimplemented from ParticleZoo::PhaseSpaceFileWriter.

6.8.3.7 canHaveConstantZ()

`bool ParticleZoo::IAEAphspFile::Writer::canHaveConstantZ ( ) const [inline], [override], [protected], [virtual]`

Check if constant Z coordinates are supported.

Returns

true (IAEA format supports constant Z values)

Reimplemented from ParticleZoo::PhaseSpaceFileWriter.

6.8.3.8 fixedValuesHaveChanged()

`void ParticleZoo::IAEAphspFile::Writer::fixedValuesHaveChanged ( ) [inline], [override], [protected], [virtual]`

Handle changes to fixed/constant values.

Updates the IAEA header when constant values are modified, ensuring the header reflects the current optimization settings.

Reimplemented from ParticleZoo::PhaseSpaceFileWriter.

6.8.3.9 getFormatSpecificCLICommands()

`std::vector< CLICommand > ParticleZoo::IAEAphspFile::Writer::getFormatSpecificCLICommands ( ) [static]`

Get format-specific command-line options.

Returns

Vector of CLI commands supported by IAEA writer

6.8.3.10 getHeader()

`IAEAHeader & ParticleZoo::IAEAphspFile::Writer::getHeader ( ) [inline]`

Get access to the IAEA header for configuration.

Returns

Reference to the header

6.8.3.11 getMaximumSupportedParticles()

```
std::uint64_t ParticleZoo::IAEAphspFile::Writer::getMaximumSupportedParticles ( ) const  [inline],
[override], [virtual]
```

Get the maximum number of particles this format can store.

Returns

Maximum particle count (effectively unlimited for IAEA)

Implements ParticleZoo::PhaseSpaceFileWriter.

6.8.3.12 getParticleRecordLength()

```
std::size_t ParticleZoo::IAEAphspFile::Writer::getParticleRecordLength ( ) const  [inline], [override],
[protected], [virtual]
```

Get the length of each particle record in bytes.

Returns

Size of each particle record as configured in header

Reimplemented from ParticleZoo::PhaseSpaceFileWriter.

6.8.3.13 setNumberOfOriginalHistories()

```
void ParticleZoo::IAEAphspFile::Writer::setNumberOfOriginalHistories (
             std::uint64_t numberOfHistories )
```

Set the number of original simulation histories.

Parameters

numberOfHistories	Number of original histories to record in header

6.8.3.14 writeBinaryParticle()

```
void ParticleZoo::IAEAphspFile::Writer::writeBinaryParticle (
            ByteBuffer & buffer,
            Particle & particle )  [override], [protected], [virtual]
```

Encode and write a single particle to binary data.

Parameters

buffer	Binary buffer to write particle data to
particle	Particle object to encode and store

Exceptions

std::runtime_error	if particle type is unsupported

Reimplemented from ParticleZoo::PhaseSpaceFileWriter.

6.8.3.15 writeHeaderData()

```
void ParticleZoo::IAEAphspFile::Writer::writeHeaderData (
            ByteBuffer & buffer )  [override], [protected], [virtual]
```

Write header data to the output buffer (not used for IAEA)

Parameters

buffer	Binary buffer for header information (not used for IAEA)

Implements ParticleZoo::PhaseSpaceFileWriter.

The documentation for this class was generated from the following files:

- include/particlezoo/IAEA/IAEAphspFile.h
- src/IAEA/IAEAphspFile.cc

6.9 ParticleZoo::Particle Class Reference

Represents a particle in phase space.

```
#include <particlezoo/Particle.h>
```

Public Member Functions

- Particle ()=default

  Default constructor for Particle.
- Particle (ParticleType type, float kineticEnergy, float x, float y, float z, float directionalCosineX, float directional↩ CosineY, float directionalCosineZ, bool isNewHistory=true, float weight=1.0)

  Construct a Particle with specified properties.
- bool getBoolProperty (BoolPropertyType type) const

  Get the value of a boolean property.
- const std::vector< bool > & getCustomBoolProperties () const

  Get a reference to all custom boolean properties.
- const std::vector< float > & getCustomFloatProperties () const

  Get a reference to all custom float properties.
- const std::vector< std::int32_t > & getCustomIntProperties () const

  Get a reference to all custom integer properties.
- const std::vector< std::string > & getCustomStringProperties () const

  Get a reference to all custom string properties.
- float getDirectionalCosineX () const

  Get the X component of the directional cosine (momentum unit vector).
- float getDirectionalCosineY () const

  Get the Y component of the directional cosine (momentum unit vector).
- float getDirectionalCosineZ () const

  Get the Z component of the directional cosine (momentum unit vector).
- float getFloatProperty (FloatPropertyType type) const

  Get the value of a float property.
- std::uint32_t getIncrementalHistories () const

  Convenience function to get the number of incremental histories regardless of whether the property is set. If the property is not set, it returns 1 if the particle is marked as a new history, otherwise 0.
- std::int32_t getIntProperty (IntPropertyType type) const

  Get the value of an integer property.
- float getKineticEnergy () const

  Get the kinetic energy of the particle.
- int getNumberOfBoolProperties () const

  Get the number of boolean properties currently stored.
- int getNumberOfFloatProperties () const

Get the number of float properties currently stored.

- int getNumberOfIntProperties () const

  Get the number of integer properties currently stored.
- ParticleType getType () const

  Get the particle type.
- float getWeight () const

  Get the statistical weight of the particle.
- float getX () const

  Get the X coordinate position of the particle.
- float getY () const

  Get the Y coordinate position of the particle.
- float getZ () const

  Get the Z coordinate position of the particle.
- bool hasBoolProperty (BoolPropertyType type) const

  Check if a boolean property of the specified type exists.
- bool hasFloatProperty (FloatPropertyType type) const

  Check if a float property of the specified type exists.
- bool hasIntProperty (IntPropertyType type) const

  Check if an integer property of the specified type exists.
- bool isNewHistory () const

  Check if this particle starts a new Monte Carlo history.
- bool projectToXValue (float X)

  Project the particle's trajectory to a specific X coordinate.
- bool projectToYValue (float Y)

  Project the particle's trajectory to a specific Y coordinate.
- bool projectToZValue (float Z)

  Project the particle's trajectory to a specific Z coordinate.
- void reserveBoolProperties (unsigned int size)

  Reserve memory for boolean properties.
- void reserveFloatProperties (unsigned int size)

  Reserve memory for float properties.
- void reserveIntProperties (unsigned int size)

  Reserve memory for integer properties.
- void setBoolProperty (BoolPropertyType type, bool value)

  Set the value of a boolean property.
- void setDirectionalCosineX (float px)

  Set the X component of the directional cosine (momentum unit vector).
- void setDirectionalCosineY (float py)

  Set the Y component of the directional cosine (momentum unit vector).
- void setDirectionalCosineZ (float pz)

  Set the Z component of the directional cosine (momentum unit vector).

- void setFloatProperty (FloatPropertyType type, float value)

  Set the value of a float property.
- void setIncrementalHistories (std::uint32_t incrementalHistories)

  Convenience function to set the number of incremental histories using the INCREMENTAL_HISTORY_NUMBER integer property.
- void setIntProperty (IntPropertyType type, std::int32_t value)

  Set the value of an integer property.
- void setKineticEnergy (float energy)

  Set the kinetic energy of the particle.
- void setNewHistory (bool isNewHistory)

  Set whether this particle starts a new Monte Carlo history.
- void setStringProperty (std::string value)

  Add a custom string property.
- void setWeight (float weight)

  Set the statistical weight of the particle.
- void setX (float x)

  Set the X coordinate position of the particle.
- void setY (float y)

  Set the Y coordinate position of the particle.
- void setZ (float z)

  Set the Z coordinate position of the particle.

6.9.1 Detailed Description

Represents a particle in phase space.

The Particle class encapsulates all the information about a single particle including its position, momentum direction, kinetic energy, statistical weight, and additional properties specific to different simulation codes. It provides methods for manipulating particle properties, projecting particle trajectories, and storing format-specific metadata.

6.9.2 Constructor & Destructor Documentation

6.9.2.1 Particle() [1/2]

```
ParticleZoo::Particle::Particle ( )  [default]
```

Default constructor for Particle.

Creates a particle with default values (unsupported type, zero energy, etc.).

6.9.2.2 Particle() [2/2]

```
ParticleZoo::Particle::Particle (
             ParticleType type,
             float kineticEnergy,
             float x,
             float y,
             float z,
             float directionalCosineX,
             float directionalCosineY,
             float directionalCosineZ,
             bool isNewHistory = true,
             float weight = 1.0 )  [inline]
```

Construct a Particle with specified properties.

Creates a particle with the given position, momentum direction, energy, and other properties. The directional cosines are automatically normalized to ensure they represent a unit vector.

Parameters

type	The particle type (electron, photon, proton, etc.)
kineticEnergy	The kinetic energy of the particle
x	The X coordinate position
y	The Y coordinate position
z	The Z coordinate position
directionalCosineX	The X component of the momentum unit vector
directionalCosineY	The Y component of the momentum unit vector
directionalCosineZ	The Z component of the momentum unit vector
isNewHistory	Whether this particle starts a new Monte Carlo history (default: true)
weight	The statistical weight of the particle (default: 1.0)

6.9.3 Member Function Documentation

6.9.3.1 getBoolProperty()

```
bool ParticleZoo::Particle::getBoolProperty (
             BoolPropertyType type ) const  [inline]
```

Get the value of a boolean property.

Parameters

type	The boolean property type to retrieve

Returns

bool The value of the boolean property

Exceptions

std::invalid_argument	if the property type is invalid or not found

6.9.3.2 getCustomBoolProperties()

```
const std::vector< bool > & ParticleZoo::Particle::getCustomBoolProperties ( ) const  [inline]
```

Get a reference to all custom boolean properties.

Returns

const std::vector<bool>& Reference to the vector of custom boolean properties

6.9.3.3 getCustomFloatProperties()

```
const std::vector< float > & ParticleZoo::Particle::getCustomFloatProperties ( ) const  [inline]
```

Get a reference to all custom float properties.

Returns

const std::vector<float>& Reference to the vector of custom float properties

6.9.3.4 getCustomIntProperties()

```
const std::vector< std::int32_t > & ParticleZoo::Particle::getCustomIntProperties ( ) const  [inline]
```

Get a reference to all custom integer properties.

Returns

const std::vector<std::int32_t>& Reference to the vector of custom integer properties

6.9.3.5 getCustomStringProperties()

`const std::vector< std::string > & ParticleZoo::Particle::getCustomStringProperties ( ) const [inline]`

Get a reference to all custom string properties.

Returns

const std::vector<std::string>& Reference to the vector of custom string properties

6.9.3.6 getDirectionalCosineX()

`float ParticleZoo::Particle::getDirectionalCosineX ( ) const [inline]`

Get the X component of the directional cosine (momentum unit vector).

Returns

float The X component of the directional cosine

6.9.3.7 getDirectionalCosineY()

`float ParticleZoo::Particle::getDirectionalCosineY ( ) const [inline]`

Get the Y component of the directional cosine (momentum unit vector).

Returns

float The Y component of the directional cosine

6.9.3.8 getDirectionalCosineZ()

`float ParticleZoo::Particle::getDirectionalCosineZ ( ) const [inline]`

Get the Z component of the directional cosine (momentum unit vector).

Returns

float The Z component of the directional cosine

6.9.3.9 getFloatProperty()

```
float ParticleZoo::Particle::getFloatProperty (
            FloatPropertyType type ) const  [inline]
```

Get the value of a float property.

Parameters

type	The float property type to retrieve

Returns

float The value of the float property

Exceptions

std::invalid_argument	if the property type is invalid or not found

6.9.3.10 getIncrementalHistories()

```
std::uint32_t ParticleZoo::Particle::getIncrementalHistories ( ) const  [inline]
```

Convenience function to get the number of incremental histories regardless of whether the property is set. If the property is not set, it returns 1 if the particle is marked as a new history, otherwise 0.

Returns

std::uint32_t The number of incremental histories

6.9.3.11 getIntProperty()

```
std::int32_t ParticleZoo::Particle::getIntProperty (
             IntPropertyType type ) const  [inline]
```

Get the value of an integer property.

Parameters

type	The integer property type to retrieve

Returns

std::int32_t The value of the integer property

Exceptions

std::invalid_argument	if the property type is invalid or not found

6.9.3.12 getKineticEnergy()

```
float ParticleZoo::Particle::getKineticEnergy ( ) const  [inline]
```

Get the kinetic energy of the particle.

Returns

float The kinetic energy value

6.9.3.13 getNumberOfBoolProperties()

```
int ParticleZoo::Particle::getNumberOfBoolProperties ( ) const  [inline]
```

Get the number of boolean properties currently stored.

Returns

int The number of boolean properties

6.9.3.14 getNumberOfFloatProperties()

```
int ParticleZoo::Particle::getNumberOfFloatProperties ( ) const  [inline]
```

Get the number of float properties currently stored.

Returns

int The number of float properties

6.9.3.15 getNumberOfIntProperties()

```
int ParticleZoo::Particle::getNumberOfIntProperties ( ) const  [inline]
```

Get the number of integer properties currently stored.

Returns

int The number of integer properties

6.9.3.16 getType()

```
ParticleType ParticleZoo::Particle::getType ( ) const  [inline]
```

Get the particle type.

Returns

ParticleType The type of particle (electron, photon, proton, etc.)

6.9.3.17 getWeight()

```
float ParticleZoo::Particle::getWeight ( ) const  [inline]
```

Get the statistical weight of the particle.

Returns

float The statistical weight value

6.9.3.18 getX()

```
float ParticleZoo::Particle::getX ( ) const  [inline]
```

Get the X coordinate position of the particle.

Returns

float The X coordinate value

6.9.3.19 getY()

```
float ParticleZoo::Particle::getY ( ) const  [inline]
```

Get the Y coordinate position of the particle.

Returns

float The Y coordinate value

6.9.3.20 getZ()

```
float ParticleZoo::Particle::getZ ( ) const  [inline]
```

Get the Z coordinate position of the particle.

Returns

float The Z coordinate value

6.9.3.21 hasBoolProperty()

```
bool ParticleZoo::Particle::hasBoolProperty (
             BoolPropertyType type ) const  [inline]
```

Check if a boolean property of the specified type exists.

Parameters

type	The boolean property type to check for

Returns

true if the property exists

false if the property does not exist

6.9.3.22 hasFloatProperty()

```
bool ParticleZoo::Particle::hasFloatProperty (
            FloatPropertyType type ) const  [inline]
```

Check if a float property of the specified type exists.

Parameters

type	The float property type to check for

Returns

true if the property exists

false if the property does not exist

6.9.3.23 hasIntProperty()

```
bool ParticleZoo::Particle::hasIntProperty (
            IntPropertyType type ) const  [inline]
```

Check if an integer property of the specified type exists.

Parameters

type	The integer property type to check for

Returns

true if the property exists

false if the property does not exist

6.9.3.24 isNewHistory()

```
bool ParticleZoo::Particle::isNewHistory ( ) const  [inline]
```

Check if this particle starts a new Monte Carlo history.

Returns

true if this particle starts a new history

false if this particle continues an existing history

6.9.3.25 projectToXValue()

```
bool ParticleZoo::Particle::projectToXValue (
            float X ) [inline]
```

Project the particle's trajectory to a specific X coordinate.

Calculates where the particle would be when it reaches the specified X value, assuming it travels in a straight line. Updates the Y and Z coordinates accordingly.

Parameters

X	The target X coordinate to project to

Returns

true if projection was successful

false if projection is impossible (particle has no movement in X direction)

6.9.3.26 projectToYValue()

```
bool ParticleZoo::Particle::projectToYValue (
            float Y ) [inline]
```

Project the particle's trajectory to a specific Y coordinate.

Calculates where the particle would be when it reaches the specified Y value, assuming it travels in a straight line. Updates the X and Z coordinates accordingly.

Parameters

Y	The target Y coordinate to project to

Returns

true if projection was successful

false if projection is impossible (particle has no movement in Y direction)

6.9.3.27 projectToZValue()

```
bool ParticleZoo::Particle::projectToZValue (
            float Z )  [inline]
```

Project the particle's trajectory to a specific Z coordinate.

Calculates where the particle would be when it reaches the specified Z value, assuming it travels in a straight line. Updates the X and Y coordinates accordingly.

Parameters

Z	The target Z coordinate to project to

Returns

true if projection was successful

false if projection is impossible (particle has no movement in Z direction)

6.9.3.28 reserveBoolProperties()

```
void ParticleZoo::Particle::reserveBoolProperties (
            unsigned int size )  [inline]
```

Reserve memory for boolean properties.

Parameters

size	The number of boolean properties to reserve space for

6.9.3.29 reserveFloatProperties()

```
void ParticleZoo::Particle::reserveFloatProperties (
            unsigned int size )  [inline]
```

Reserve memory for float properties.

Parameters

size	The number of float properties to reserve space for

6.9.3.30 reserveIntProperties()

```
void ParticleZoo::Particle::reserveIntProperties (
             unsigned int size )  [inline]
```

Reserve memory for integer properties.

Parameters

size	The number of integer properties to reserve space for

6.9.3.31 setBoolProperty()

```
void ParticleZoo::Particle::setBoolProperty (
             BoolPropertyType type,
             bool value )  [inline]
```

Set the value of a boolean property.

If the property doesn't exist, it will be created. If it exists, the value will be updated.

Parameters

type	The boolean property type to set
value	The value to set for the property

6.9.3.32 setDirectionalCosineX()

```
void ParticleZoo::Particle::setDirectionalCosineX (
             float px )  [inline]
```

Set the X component of the directional cosine (momentum unit vector).

Parameters

px	The X component of the directional cosine to set

6.9.3.33 setDirectionalCosineY()

```
void ParticleZoo::Particle::setDirectionalCosineY (
             float py )  [inline]
```

Set the Y component of the directional cosine (momentum unit vector).

Parameters

py	The Y component of the directional cosine to set

6.9.3.34 setDirectionalCosineZ()

```
void ParticleZoo::Particle::setDirectionalCosineZ (
             float pz )  [inline]
```

Set the Z component of the directional cosine (momentum unit vector).

Parameters

pz	The Z component of the directional cosine to set

6.9.3.35 setFloatProperty()

```
void ParticleZoo::Particle::setFloatProperty (
             FloatPropertyType type,
             float value )  [inline]
```

Set the value of a float property.

If the property doesn't exist, it will be created. If it exists, the value will be updated.

Parameters

type	The float property type to set
value	The value to set for the property

6.9.3.36 setIncrementalHistories()

```
void ParticleZoo::Particle::setIncrementalHistories (
            std::uint32_t incrementalHistories )  [inline]
```

Convenience function to set the number of incremental histories using the INCREMENTAL_HISTORY_NUMBER integer property.

Parameters

incrementalHistories	The number of incremental histories to set (must be greater than 0)

6.9.3.37 setIntProperty()

```
void ParticleZoo::Particle::setIntProperty (
            IntPropertyType type,
            std::int32_t value )  [inline]
```

Set the value of an integer property.

If the property doesn't exist, it will be created. If it exists, the value will be updated.

Parameters

type	The integer property type to set
value	The value to set for the property

6.9.3.38 setKineticEnergy()

```
void ParticleZoo::Particle::setKineticEnergy (
            float energy )  [inline]
```

Set the kinetic energy of the particle.

Parameters

energy	The kinetic energy value to set

6.9.3.39 setNewHistory()

```
void ParticleZoo::Particle::setNewHistory (
            bool isNewHistory )  [inline]
```

Set whether this particle starts a new Monte Carlo history.

Parameters

isNewHistory	True if this particle starts a new history, false otherwise

6.9.3.40 setStringProperty()

```
void ParticleZoo::Particle::setStringProperty (
            std::string value )  [inline]
```

Add a custom string property.

Associate a string value with this particle. Multiple string properties can be added.

Parameters

value	The string value to add as a property

6.9.3.41 setWeight()

```
void ParticleZoo::Particle::setWeight (
            float weight )  [inline]
```

Set the statistical weight of the particle.

Parameters

weight	The statistical weight value to set

6.9.3.42 setX()

```
void ParticleZoo::Particle::setX (
            float x )  [inline]
```

Set the X coordinate position of the particle.

Parameters

x	The X coordinate value to set

6.9.3.43 setY()

```
void ParticleZoo::Particle::setY (
            float y )  [inline]
```

Set the Y coordinate position of the particle.

Parameters

y	The Y coordinate value to set

6.9.3.44 setZ()

```
void ParticleZoo::Particle::setZ (
            float z )  [inline]
```

Set the Z coordinate position of the particle.

Parameters

z	The Z coordinate value to set

The documentation for this class was generated from the following file:

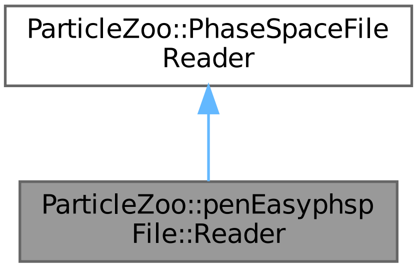
- include/particlezoo/Particle.h

6.10 ParticleZoo::penEasyphspFile::Reader Class Reference

Reader for penEasy format phase space files.

```
#include <particlezoo/peneasy/penEasyphspFile.h>
```

Inheritance diagram for ParticleZoo::penEasyphspFile::Reader:


ParticleZoo::PhaseSpaceFile Reader
ParticleZoo::penEasyphsp File::Reader

Public Member Functions

- Reader (const std::string &fileName, const UserOptions &options=UserOptions{})

  Construct reader for penEasy phase space file.
- std::uint64_t getNumberOfOriginalHistories () const override

  Get the number of original simulation histories.
- std::uint64_t getNumberOfParticles () const override

  Get the total number of particles in the phase space.

Public Member Functions inherited from ParticleZoo::PhaseSpaceFileReader

- PhaseSpaceFileReader (const std::string &phspFormat, const std::string &fileName, const UserOptions &user↩ Options, FormatType formatType=FormatType::BINARY, const FixedValues fixedValues=FixedValues(), unsigned int bufferSize=DEFAULT_BUFFER_SIZE)

  Construct a new Phase Space File Reader object.
- virtual ∼PhaseSpaceFileReader ()

  Destroy the Phase Space File Reader object.
- void close ()

Close the phase space file and clean up resources.
- float getConstantPx () const

  Get the constant X-component of the direction unit vector (if constant).
- float getConstantPy () const

  Get the constant Y-component of the direction unit vector (if constant).
- float getConstantPz () const

  Get the constant Z-component of the direction unit vector (if constant).
- float getConstantWeight () const

  Get the constant statistical weight value (if constant).
- float getConstantX () const

  Get the constant X coordinate value (if constant).
- float getConstantY () const

  Get the constant Y coordinate value (if constant).
- float getConstantZ () const

  Get the constant Z coordinate value (if constant).
- const std::string getFileName () const

  Get the filename of the phase space file being read.
- std::uint64_t getFileSize () const

  Get the size of the phase space file in bytes.
- const FixedValues getFixedValues () const

  Get the fixed values configuration.
- virtual std::uint64_t getHistoriesRead ()

  Get the number of Monte Carlo histories that have been read so far. If the end of the file has been reached, this will return the total number of original histories unless more histories than expected have already been read - in which case it returns the actual count.
- Particle getNextParticle ()

  Get the next particle from the phase space file.
- virtual std::uint64_t getParticlesRead ()

  Get the number of particles that have been read so far.
- const std::string getPHSPFormat () const

  Get the phase space file format identifier.
- virtual bool hasMoreParticles ()

  Check if there are more particles to read in the file.
- bool isPxConstant () const

  Check if the X-component of momentum is constant for all particles.
- bool isPyConstant () const

  Check if the Y-component of momentum is constant for all particles.
- bool isPzConstant () const

  Check if the Z-component of momentum is constant for all particles.
- bool isWeightConstant () const

  Check if the statistical weight is constant for all particles.
- bool isXConstant () const

Check if the X coordinate is constant for all particles.
- bool isYConstant () const

  Check if the Y coordinate is constant for all particles.
- bool isZConstant () const

  Check if the Z coordinate is constant for all particles.
- void moveToParticle (std::uint64_t particleIndex)

  Move the file position to a specific particle index.
- void setCommentMarkers (const std::vector< std::string > &commentMarkers)

  Set comment markers for ASCII format files.

Protected Member Functions

- size_t getMaximumASCIILineLength () const override

  Get the maximum length of ASCII particle lines, required for buffer sizing.
- Particle readASCIIParticle (const std::string &line) override

  Parse a single ASCII line into a Particle object.

Protected Member Functions inherited from ParticleZoo::PhaseSpaceFileReader

- double calcThirdUnitComponent (double &u, double &v) const

  Calculate the third component of a unit vector from two components (double precision).
- float calcThirdUnitComponent (float &u, float &v) const

  Calculate the third component of a unit vector from two components (float precision).
- const ByteBuffer getHeaderData ()

  Get the file header data as a byte buffer.
- const ByteBuffer getHeaderData (std::size_t headerSize)

  Get a specific amount of header data as a byte buffer.
- Particle getNextParticle (bool countParticleInStatistics)

  Get the next particle with optional statistics counting control.
- std::size_t getNumberOfEntriesInFile () const

  Get the number of particle records that fit in the file.
- virtual std::size_t getParticleRecordLength () const

  Get the length in bytes of each particle record.
- virtual std::size_t getParticleRecordStartOffset () const

  Get the byte offset where particle records start in the file.
- virtual std::uint64_t getParticlesRead (bool includeSkippedParticles)

  Get the number of particles read with optional inclusion of skipped particles (including pseudo-particles).
- const UserOptions & getUserOptions () const

  Get the user options that were passed to the constructor.
- virtual Particle readBinaryParticle (ByteBuffer &buffer)

Read a particle from binary data.
- virtual Particle readParticleManually ()

  Read a particle manually (for formats requiring third-party I/O).
- void setByteOrder (ByteOrder byteOrder)

  Set the byte order for binary data interpretation.
- void setConstantPx (float Px)

  Set a constant X-component of the direction unit vector for all particles.
- void setConstantPy (float Py)

  Set a constant Y-component of the direction unit vector for all particles.
- void setConstantPz (float Pz)

  Set a constant Z-component of the direction unit vector for all particles.
- void setConstantWeight (float weight)

  Set a constant statistical weight for all particles.
- void setConstantX (float X)

  Set a constant X coordinate value for all particles.
- void setConstantY (float Y)

  Set a constant Y coordinate value for all particles.
- void setConstantZ (float Z)

  Set a constant Z coordinate value for all particles.

Additional Inherited Members

Static Public Member Functions inherited from ParticleZoo::PhaseSpaceFileReader

- static std::vector< CLICommand > getCLICommands ()

  Get command line interface commands supported by this reader.

6.10.1 Detailed Description

Reader for penEasy format phase space files.

Provides functionality to read phase space data from penEasy ASCII format files. Automatically counts total particles and total histories during construction by scanning the entire file (this may be slow for very large files).

6.10.2 Constructor & Destructor Documentation

6.10.2.1 Reader()

```
ParticleZoo::penEasyphspFile::Reader::Reader (
             const std::string & fileName,
             const UserOptions & options = UserOptions{} )
```

Construct reader for penEasy phase space file.

Scans the file during construction to count particles and sum delta-N values for determining the total number of original histories.

Parameters

fileName	Path to the penEasy phase space file to read
options	User-specified options for reading behavior

Exceptions

std::runtime_error	if file cannot be opened or parsed

6.10.3 Member Function Documentation

6.10.3.1 getMaximumASCIILineLength()

```
size_t ParticleZoo::penEasyphspFile::Reader::getMaximumASCIILineLength ( ) const  [inline], [override],
[protected], [virtual]
```

Get the maximum length of ASCII particle lines, required for buffer sizing.

Returns

Maximum line length

Reimplemented from ParticleZoo::PhaseSpaceFileReader.

6.10.3.2 getNumberOfOriginalHistories()

```
std::uint64_t ParticleZoo::penEasyphspFile::Reader::getNumberOfOriginalHistories ( ) const  [inline],
[override], [virtual]
```

Get the number of original simulation histories.

Returns

Sum of all delta-N values from particle records

Implements ParticleZoo::PhaseSpaceFileReader.

6.10.3.3 getNumberOfParticles()

```
std::uint64_t ParticleZoo::penEasyphspFile::Reader::getNumberOfParticles ( ) const  [inline],
[override], [virtual]
```

Get the total number of particles in the phase space.

Returns

Total particle count determined by file scanning

Implements ParticleZoo::PhaseSpaceFileReader.

6.10.3.4 readASCIIParticle()

```
Particle ParticleZoo::penEasyphspFile::Reader::readASCIIParticle (
             const std::string & line )  [override], [protected], [virtual]
```

Parse a single ASCII line into a Particle object.

Parses penEasy format: KPAR E X Y Z U V W WGHT DeltaN ILB(1..5)

- KPAR: particle type code (1=electron, 2=photon, 3=positron, 4=proton)
- E: kinetic energy in eV
- X,Y,Z: position coordinates
- U,V,W: direction cosines
- WGHT: particle weight
- DeltaN: incremental history number
- ILB(1..5): PENELOPE ILB array values

Parameters

line	ASCII line containing particle data

Returns

Parsed Particle object with all properties set

Exceptions

std::runtime_error	if line cannot be parsed or contains invalid data

Reimplemented from ParticleZoo::PhaseSpaceFileReader.

The documentation for this class was generated from the following files:

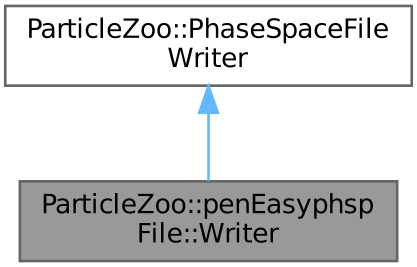
- include/particlezoo/peneasy/penEasyphspFile.h
- src/peneasy/penEasyphspFile.cc

6.11 ParticleZoo::penEasyphspFile::Writer Class Reference

Writer for penEasy format phase space files.

```
#include <particlezoo/peneasy/penEasyphspFile.h>
```

Inheritance diagram for ParticleZoo::penEasyphspFile::Writer:


ParticleZoo::PhaseSpaceFile Writer
ParticleZoo::penEasyphsp File::Writer

Public Member Functions

- Writer (const std::string &fileName, const UserOptions &options=UserOptions{})

  Construct writer for penEasy phase space file.
- std::uint64_t getMaximumSupportedParticles () const override

  Get the maximum number of particles this format can store.

Public Member Functions inherited from ParticleZoo::PhaseSpaceFileWriter

- PhaseSpaceFileWriter (const std::string &phspFormat, const std::string &fileName, const UserOptions &user↩ Options, FormatType formatType=FormatType::BINARY, const FixedValues fixedValues=FixedValues(), unsigned int bufferSize=DEFAULT_BUFFER_SIZE)

  Construct a new Phase Space File Writer object.
- virtual ∼PhaseSpaceFileWriter ()

  Destroy the Phase Space File Writer object.
- void addAdditionalHistories (std::uint64_t additionalHistories)

  Add additional Monte Carlo histories to the count.
- void close ()

Close the phase space file and finalize writing.
- ByteOrder getByteOrder () const

  Get the byte order used for binary data writing.
- float getConstantPx () const

  Get the constant X-component of the direction unit vector (if constant).
- float getConstantPy () const

  Get the constant Y-component of the direction unit vector (if constant).
- float getConstantPz () const

  Get the constant Z-component of the direction unit vector (if constant).
- float getConstantWeight () const

  Get the constant statistical weight value (if constant).
- float getConstantX () const

  Get the constant X coordinate value (if constant).
- float getConstantY () const

  Get the constant Y coordinate value (if constant).
- float getConstantZ () const

  Get the constant Z coordinate value (if constant).
- const std::string getFileName () const

  Get the filename where the phase space file is being written.
- const FixedValues getFixedValues () const

  Get the fixed values configuration.
- virtual std::uint64_t getHistoriesWritten () const

  Get the number of Monte Carlo histories that have been written.
- std::uint64_t getParticlesWritten () const

  Get the number of particles that have been written to the file.
- const std::string getPHSPFormat () const

  Get the phase space file format identifier.
- bool isPxConstant () const

  Check if the X-component of the direction unit vector is set to a constant value for all particles.
- bool isPyConstant () const

  Check if the Y-component of the direction unit vector is set to a constant value for all particles.
- bool isPzConstant () const

  Check if the Z-component of the direction unit vector is set to a constant value for all particles.
- bool isWeightConstant () const

  Check if the statistical weight is set to a constant value for all particles.
- bool isXConstant () const

  Check if the X coordinate is set to a constant value for all particles.
- bool isYConstant () const

  Check if the Y coordinate is set to a constant value for all particles.
- bool isZConstant () const

  Check if the Z coordinate is set to a constant value for all particles.
- virtual void writeParticle (Particle particle)

  Write a particle to the phase space file.

Protected Member Functions

- size_t getMaximumASCIILineLength () const override

  Get the maximum length of ASCII particle lines, required for buffer sizing.
- std::size_t getParticleRecordStartOffset () const override

  Get the byte offset where particle records start.
- const std::string writeASCIIParticle (Particle &particle) override

  Convert a particle to ASCII representation.
- void writeHeaderData (ByteBuffer &buffer) override

  Write the file header to the output buffer.

Protected Member Functions inherited from ParticleZoo::PhaseSpaceFileWriter

- virtual bool accountForAdditionalHistories (std::uint64_t additionalHistories)

  Handle accounting for simulation histories that produced no particles.
- virtual bool canHaveConstantPx () const

  Check if this format supports constant X-component of the direction unit vector.
- virtual bool canHaveConstantPy () const

  Check if this format supports constant Y-component of the direction unit vector.
- virtual bool canHaveConstantPz () const

  Check if this format supports constant Z-component of the direction unit vector.
- virtual bool canHaveConstantWeight () const

  Check if this format supports constant statistical weights.
- virtual bool canHaveConstantX () const

  Check if this format supports constant X coordinates.
- virtual bool canHaveConstantY () const

  Check if this format supports constant Y coordinates.
- virtual bool canHaveConstantZ () const

  Check if this format supports constant Z coordinates.
- virtual bool canWritePseudoParticlesExplicitly () const

  Check if this format can write pseudo-particles explicitly.
- virtual void fixedValuesHaveChanged ()

  Called when fixed values have been changed.
- virtual std::size_t getParticleRecordLength () const

  Get the length in bytes of each particle record.
- virtual std::uint64_t getPendingHistories () const

  Get the number of pending histories to account for.
- const UserOptions & getUserOptions () const

  Get the user options that were passed to the constructor.
- void setByteOrder (ByteOrder byteOrder)

  Set the byte order for binary data writing.

- void setConstantPx (float Px)

  Set a constant X-component of the direction unit vector for all particles.
- void setConstantPy (float Py)

  Set a constant Y-component of the direction unit vector for all particles.
- void setConstantPz (float Pz)

  Set a constant Z-component of the direction unit vector for all particles.
- void setConstantWeight (float weight)

  Set a constant statistical weight for all particles.
- void setConstantX (float X)

  Set a constant X coordinate value for all particles.
- void setConstantY (float Y)

  Set a constant Y coordinate value for all particles.
- void setConstantZ (float Z)

  Set a constant Z coordinate value for all particles.
- virtual void writeBinaryParticle (ByteBuffer &buffer, Particle &particle)

  Write a particle in binary format to a byte buffer.
- virtual void writeParticleManually (Particle &particle)

  Write a particle manually (for formats requiring third-party I/O).

Additional Inherited Members

Static Public Member Functions inherited from ParticleZoo::PhaseSpaceFileWriter

- static std::vector< CLICommand > getCLICommands ()

  Get command line interface commands supported by this writer.

6.11.1 Detailed Description

Writer for penEasy format phase space files.

Provides functionality to write phase space data in the penEasy ASCII format, which is compatible with the PENELOPE Monte Carlo transport code. The format includes particle type, energy, position, direction, weight, and PENELOPE-specific values (ILB1-ILB5) and delta-N values.

6.11.2 Constructor & Destructor Documentation

6.11.2.1 Writer()

```
ParticleZoo::penEasyphspFile::Writer::Writer (
                const std::string & fileName,
                const UserOptions & options = UserOptions{} )
```

Construct writer for penEasy phase space file.

Parameters

fileName	Path to the output penEasy phase space file
options	User-specified options for writing behavior

6.11.3 Member Function Documentation

6.11.3.1 getMaximumASCIILineLength()

`size_t ParticleZoo::penEasyphspFile::Writer::getMaximumASCIILineLength ( ) const [inline], [override], [protected], [virtual]`

Get the maximum length of ASCII particle lines, required for buffer sizing.

Returns

Maximum line length

Reimplemented from ParticleZoo::PhaseSpaceFileWriter.

6.11.3.2 getMaximumSupportedParticles()

`std::uint64_t ParticleZoo::penEasyphspFile::Writer::getMaximumSupportedParticles ( ) const [inline], [override], [virtual]`

Get the maximum number of particles this format can store.

Returns

Maximum particle count

Implements ParticleZoo::PhaseSpaceFileWriter.

6.11.3.3 getParticleRecordStartOffset()

`std::size_t ParticleZoo::penEasyphspFile::Writer::getParticleRecordStartOffset ( ) const [inline], [override], [protected], [virtual]`

Get the byte offset where particle records start.

Returns

Header length (112 bytes for penEasy format)

Reimplemented from ParticleZoo::PhaseSpaceFileWriter.

6.11.3.4 writeASCIIParticle()

```
const std::string ParticleZoo::penEasyphspFile::Writer::writeASCIIParticle (
             Particle & particle )  [override], [protected], [virtual]
```

Convert a particle to ASCII representation.

Formats a particle according to the penEasy specification: KPAR E X Y Z U V W WGHT DeltaN ILB(1..5)

Parameters

particle	Particle object to convert to ASCII

Returns

ASCII string representation of the particle

Exceptions

std::runtime_error	if particle type is unsupported or data is too long

Reimplemented from ParticleZoo::PhaseSpaceFileWriter.

6.11.3.5 writeHeaderData()

```
void ParticleZoo::penEasyphspFile::Writer::writeHeaderData (
             ByteBuffer & buffer )  [override], [protected], [virtual]
```

Write the file header to the output buffer.

Parameters

buffer	Byte buffer to write header data to

Implements ParticleZoo::PhaseSpaceFileWriter.

The documentation for this class was generated from the following files:

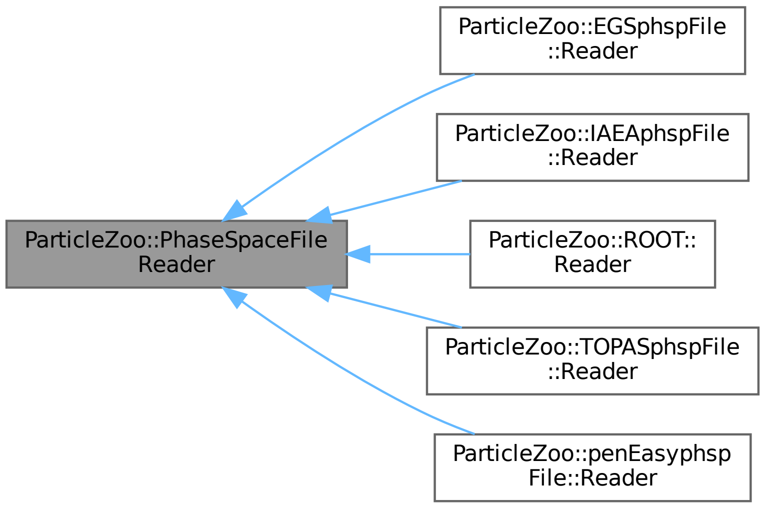
- include/particlezoo/peneasy/penEasyphspFile.h
- src/peneasy/penEasyphspFile.cc

6.12 ParticleZoo::PhaseSpaceFileReader Class Reference

Base class for reading phase space files.

```
#include <particlezoo/PhaseSpaceFileReader.h>
```

Inheritance diagram for ParticleZoo::PhaseSpaceFileReader:


ParticleZoo::EGSphspFile ::Reader
ParticleZoo::IAEAphspFile ::Reader
ParticleZoo::PhaseSpaceFile Reader
ParticleZoo::ROOT:: Reader
ParticleZoo::TOPASphspFile ::Reader
ParticleZoo::penEasyphsp File::Reader

Public Member Functions

- PhaseSpaceFileReader (const std::string &phspFormat, const std::string &fileName, const UserOptions &user←
Options, FormatType formatType=FormatType::BINARY, const FixedValues fixedValues=FixedValues(), unsigned int bufferSize=DEFAULT_BUFFER_SIZE)

  Construct a new Phase Space File Reader object.
- virtual ∼PhaseSpaceFileReader ()

  Destroy the Phase Space File Reader object.
- void close ()

  Close the phase space file and clean up resources.
- float getConstantPx () const

  Get the constant X-component of the direction unit vector (if constant).
- float getConstantPy () const

Get the constant Y-component of the direction unit vector (if constant).
- float getConstantPz () const

  Get the constant Z-component of the direction unit vector (if constant).
- float getConstantWeight () const

  Get the constant statistical weight value (if constant).
- float getConstantX () const

  Get the constant X coordinate value (if constant).
- float getConstantY () const

  Get the constant Y coordinate value (if constant).
- float getConstantZ () const

  Get the constant Z coordinate value (if constant).
- const std::string getFileName () const

  Get the filename of the phase space file being read.
- std::uint64_t getFileSize () const

  Get the size of the phase space file in bytes.
- const FixedValues getFixedValues () const

  Get the fixed values configuration.
- virtual std::uint64_t getHistoriesRead ()

  Get the number of Monte Carlo histories that have been read so far. If the end of the file has been reached, this will return the total number of original histories unless more histories than expected have already been read - in which case it returns the actual count.
- Particle getNextParticle ()

  Get the next particle from the phase space file.
- virtual std::uint64_t getNumberOfOriginalHistories () const =0

  Get the number of original Monte Carlo histories that generated this phase space.
- virtual std::uint64_t getNumberOfParticles () const =0

  Get the total number of particles in the phase space file.
- virtual std::uint64_t getParticlesRead ()

  Get the number of particles that have been read so far.
- const std::string getPHSPFormat () const

  Get the phase space file format identifier.
- virtual bool hasMoreParticles ()

  Check if there are more particles to read in the file.
- bool isPxConstant () const

  Check if the X-component of momentum is constant for all particles.
- bool isPyConstant () const

  Check if the Y-component of momentum is constant for all particles.
- bool isPzConstant () const

  Check if the Z-component of momentum is constant for all particles.
- bool isWeightConstant () const

  Check if the statistical weight is constant for all particles.
- bool isXConstant () const

Check if the X coordinate is constant for all particles.
- bool isYConstant () const

  Check if the Y coordinate is constant for all particles.
- bool isZConstant () const

  Check if the Z coordinate is constant for all particles.
- void moveToParticle (std::uint64_t particleIndex)

  Move the file position to a specific particle index.
- void setCommentMarkers (const std::vector< std::string > &commentMarkers)

  Set comment markers for ASCII format files.

Static Public Member Functions

- static std::vector< CLICommand > getCLICommands ()

  Get command line interface commands supported by this reader.

Protected Member Functions

- double calcThirdUnitComponent (double &u, double &v) const

  Calculate the third component of a unit vector from two components (double precision).
- float calcThirdUnitComponent (float &u, float &v) const

  Calculate the third component of a unit vector from two components (float precision).
- const ByteBuffer getHeaderData ()

  Get the file header data as a byte buffer.
- const ByteBuffer getHeaderData (std::size_t headerSize)

  Get a specific amount of header data as a byte buffer.
- virtual std::size_t getMaximumASCIILineLength () const

  Get the maximum line length for ASCII format files.
- Particle getNextParticle (bool countParticleInStatistics)

  Get the next particle with optional statistics counting control.
- std::size_t getNumberOfEntriesInFile () const

  Get the number of particle records that fit in the file.
- virtual std::size_t getParticleRecordLength () const

  Get the length in bytes of each particle record.
- virtual std::size_t getParticleRecordStartOffset () const

  Get the byte offset where particle records start in the file.
- virtual std::uint64_t getParticlesRead (bool includeSkippedParticles)

  Get the number of particles read with optional inclusion of skipped particles (including pseudo-particles).
- const UserOptions & getUserOptions () const

  Get the user options that were passed to the constructor.
- virtual Particle readASCIIParticle (const std::string &line)

Read a particle from ASCII data.
- virtual Particle readBinaryParticle (ByteBuffer &buffer)

  Read a particle from binary data.
- virtual Particle readParticleManually ()

  Read a particle manually (for formats requiring third-party I/O).
- void setByteOrder (ByteOrder byteOrder)

  Set the byte order for binary data interpretation.
- void setConstantPx (float Px)

  Set a constant X-component of the direction unit vector for all particles.
- void setConstantPy (float Py)

  Set a constant Y-component of the direction unit vector for all particles.
- void setConstantPz (float Pz)

  Set a constant Z-component of the direction unit vector for all particles.
- void setConstantWeight (float weight)

  Set a constant statistical weight for all particles.
- void setConstantX (float X)

  Set a constant X coordinate value for all particles.
- void setConstantY (float Y)

  Set a constant Y coordinate value for all particles.
- void setConstantZ (float Z)

  Set a constant Z coordinate value for all particles.

6.12.1 Detailed Description

Base class for reading phase space files.

This abstract class provides a unified interface for reading particle phase space files from different simulation formats (EGS, IAEA, TOPAS, etc.). It handles both binary and ASCII file formats and provides functionality for particle iteration, statistics tracking, and format-specific optimizations. In cases where I/O must be handled by a third-party library (e.g., ROOT), this class also provides a framework for manually reading particles.

6.12.2 Constructor & Destructor Documentation

6.12.2.1 PhaseSpaceFileReader()

```
ParticleZoo::PhaseSpaceFileReader::PhaseSpaceFileReader (
             const std::string & phspFormat,
             const std::string & fileName,
             const UserOptions & userOptions,
             FormatType formatType = FormatType::BINARY,
             const FixedValues fixedValues = FixedValues(),
             unsigned int bufferSize = DEFAULT_BUFFER_SIZE )
```

Construct a new Phase Space File Reader object.

Parameters

phspFormat	The format identifier of the phase space file (e.g., "IAEA", "EGS", "TOPAS")
fileName	The path to the phase space file to read
userOptions	User-defined options for reading behavior
formatType	The format type (BINARY, ASCII, or NONE), defaults to BINARY
fixedValues	Pre-defined constant values for certain particle properties
bufferSize	Size of the internal buffer for reading, defaults to DEFAULT_BUFFER_SIZE

6.12.2.2 ∼PhaseSpaceFileReader()

```
ParticleZoo::PhaseSpaceFileReader::∼PhaseSpaceFileReader ( )  [virtual]
```

Destroy the Phase Space File Reader object.

Ensures proper cleanup of file handles and allocated resources.

6.12.3 Member Function Documentation

6.12.3.1 calcThirdUnitComponent() [1/2]

```
double ParticleZoo::PhaseSpaceFileReader::calcThirdUnitComponent (
             double & u,
             double & v ) const  [inline], [protected]
```

Calculate the third component of a unit vector from two components (double precision).

Given two components of a unit vector, calculates the third component. Handles normalization if the input components are not properly normalized.

Parameters

u	First component (may be modified for normalization)
v	Second component (may be modified for normalization)

Returns

double The calculated third component

6.12.3.2 calcThirdUnitComponent() [2/2]

```
float ParticleZoo::PhaseSpaceFileReader::calcThirdUnitComponent (
             float & u,
             float & v ) const  [inline], [protected]
```

Calculate the third component of a unit vector from two components (float precision).

Given two components of a unit vector, calculates the third component. Handles normalization if the input components are not properly normalized.

Parameters

u	First component (may be modified for normalization)
v	Second component (may be modified for normalization)

Returns

float The calculated third component

6.12.3.3 close()

```
void ParticleZoo::PhaseSpaceFileReader::close ( )
```

Close the phase space file and clean up resources.

Explicitly closes the file handle and frees associated resources. The reader cannot be used after calling this method.

6.12.3.4 getCLICommands()

```
std::vector< CLICommand > ParticleZoo::PhaseSpaceFileReader::getCLICommands ( )  [static]
```

Get command line interface commands supported by this reader.

Returns a vector of CLI commands that can be used with this reader type.

Returns

std::vector<CLICommand> Vector of supported CLI commands

6.12.3.5 getConstantPx()

```
float ParticleZoo::PhaseSpaceFileReader::getConstantPx ( ) const  [inline]
```

Get the constant X-component of the direction unit vector (if constant).

Returns

float The constant Px value

Exceptions

std::runtime_error	if Px is not constant

6.12.3.6 getConstantPy()

```
float ParticleZoo::PhaseSpaceFileReader::getConstantPy ( ) const  [inline]
```

Get the constant Y-component of the direction unit vector (if constant).

Returns

float The constant Py value

Exceptions

std::runtime_error	if Py is not constant

6.12.3.7 getConstantPz()

```
float ParticleZoo::PhaseSpaceFileReader::getConstantPz ( ) const  [inline]
```

Get the constant Z-component of the direction unit vector (if constant).

Returns

float The constant Pz value

Exceptions

std::runtime_error	if Pz is not constant

6.12.3.8 getConstantWeight()

```
float ParticleZoo::PhaseSpaceFileReader::getConstantWeight ( ) const  [inline]
```

Get the constant statistical weight value (if constant).

Returns

float The constant weight value

Exceptions

std::runtime_error	if weight is not constant

6.12.3.9 getConstantX()

```
float ParticleZoo::PhaseSpaceFileReader::getConstantX ( ) const  [inline]
```

Get the constant X coordinate value (if constant).

Returns

float The constant X coordinate value

Exceptions

std::runtime_error	if X is not constant

6.12.3.10 getConstantY()

```
float ParticleZoo::PhaseSpaceFileReader::getConstantY ( ) const  [inline]
```

Get the constant Y coordinate value (if constant).

Returns

float The constant Y coordinate value

Exceptions

std::runtime_error	if Y is not constant

6.12.3.11 getConstantZ()

```
float ParticleZoo::PhaseSpaceFileReader::getConstantZ ( ) const  [inline]
```

Get the constant Z coordinate value (if constant).

Returns

float The constant Z coordinate value

Exceptions

std::runtime_error	if Z is not constant

6.12.3.12 getFileName()

```
const std::string ParticleZoo::PhaseSpaceFileReader::getFileName ( ) const  [inline]
```

Get the filename of the phase space file being read.

Returns

const std::string The filename/path of the file

6.12.3.13 getFileSize()

```
std::uint64_t ParticleZoo::PhaseSpaceFileReader::getFileSize ( ) const  [inline]
```

Get the size of the phase space file in bytes.

Returns

std::uint64_t The file size in bytes

6.12.3.14 getFixedValues()

```
const FixedValues ParticleZoo::PhaseSpaceFileReader::getFixedValues ( ) const  [inline]
```

Get the fixed values configuration.

Returns

const FixedValues The complete fixed values structure

6.12.3.15 getHeaderData() [1/2]

```
const ByteBuffer ParticleZoo::PhaseSpaceFileReader::getHeaderData ( )  [protected]
```

Get the file header data as a byte buffer.

Reads the entire header portion of the file into a ByteBuffer. The header size is determined by getParticleRecordStartOffset().

Returns

const ByteBuffer The header data

6.12.3.16 getHeaderData() [2/2]

```
const ByteBuffer ParticleZoo::PhaseSpaceFileReader::getHeaderData (
             std::size_t headerSize )  [protected]
```

Get a specific amount of header data as a byte buffer.

Parameters

headerSize	The number of bytes to read from the header

Returns

const ByteBuffer The header data of specified size

6.12.3.17 getHistoriesRead()

```
std::uint64_t ParticleZoo::PhaseSpaceFileReader::getHistoriesRead ( )  [inline], [virtual]
```

Get the number of Monte Carlo histories that have been read so far. If the end of the file has been reached, this will return the total number of original histories unless more histories than expected have already been read - in which case it returns the actual count.

Returns

std::uint64_t The number of histories read

6.12.3.18 getMaximumASCIILineLength()

```
size_t ParticleZoo::PhaseSpaceFileReader::getMaximumASCIILineLength ( ) const  [inline], [protected],
[virtual]
```

Get the maximum line length for ASCII format files.

Must be implemented by derived classes that support ASCII format. Used for buffer allocation and parsing optimization.

Returns

std::size_t The maximum length of ASCII lines in number of characters

Exceptions

std::runtime_error	if not implemented for ASCII format

Reimplemented in ParticleZoo::penEasyphspFile::Reader, and ParticleZoo::TOPASphspFile::Reader.

6.12.3.19 getNextParticle() `[1/2]`

```
Particle ParticleZoo::PhaseSpaceFileReader::getNextParticle ( )  [inline]
```

Get the next particle from the phase space file.

Reads and returns the next particle in the file. This method automatically handles buffering and format-specific parsing. The particle is counted in the read statistics.

Returns

Particle The next particle object containing position, momentum, energy, etc.

6.12.3.20 getNextParticle() [2/2]

```
Particle ParticleZoo::PhaseSpaceFileReader::getNextParticle (
             bool countParticleInStatistics )  [protected]
```

Get the next particle with optional statistics counting control.

This protected version allows derived classes to control whether the particle should be counted in the read statistics.

Parameters

countParticleInStatistics	Whether to count this particle in statistics

Returns

Particle The next particle object

6.12.3.21 getNumberOfEntriesInFile()

```
std::size_t ParticleZoo::PhaseSpaceFileReader::getNumberOfEntriesInFile ( ) const  [inline], [protected]
```

Get the number of particle records that fit in the file.

For binary files, calculates how many complete records fit in the file. For other formats, returns getNumberOfParticles().

Returns

std::size_t The number of particle entries in the file

6.12.3.22 getNumberOfOriginalHistories()

```
virtual std::uint64_t ParticleZoo::PhaseSpaceFileReader::getNumberOfOriginalHistories ( ) const
[pure virtual]
```

Get the number of original Monte Carlo histories that generated this phase space.

This is a pure virtual method that must be implemented by derived classes as the method for determining history count varies by format.

Returns

std::uint64_t The number of original histories

Implemented in ParticleZoo::EGSphspFile::Reader, ParticleZoo::IAEAphspFile::Reader, ParticleZoo::penEasyphspFile::Reader, ParticleZoo::ROOT::Reader, and ParticleZoo::TOPASphspFile::Reader.

6.12.3.23 getNumberOfParticles()

```
virtual std::uint64_t ParticleZoo::PhaseSpaceFileReader::getNumberOfParticles ( ) const  [pure
virtual]
```

Get the total number of particles in the phase space file.

This is a pure virtual method that must be implemented by derived classes as the method for determining particle count varies by format.

Returns

std::uint64_t The total number of particles in the file

Implemented in ParticleZoo::EGSphspFile::Reader, ParticleZoo::IAEAphspFile::Reader, ParticleZoo::penEasyphspFile::Reader, ParticleZoo::ROOT::Reader, and ParticleZoo::TOPASphspFile::Reader.

6.12.3.24 getParticleRecordLength()

```
std::size_t ParticleZoo::PhaseSpaceFileReader::getParticleRecordLength ( ) const  [inline], [protected],
[virtual]
```

Get the length in bytes of each particle record.

Must be implemented by derived classes for binary formatted files.

Returns

std::size_t The length of each particle record in bytes

Exceptions

std::runtime_error	if not implemented for binary format

Reimplemented in ParticleZoo::EGSphspFile::Reader, ParticleZoo::IAEAphspFile::Reader, and ParticleZoo::TOPASphspFile::Reader.

6.12.3.25 getParticleRecordStartOffset()

```
std::size_t ParticleZoo::PhaseSpaceFileReader::getParticleRecordStartOffset ( ) const  [inline],
[protected], [virtual]
```

Get the byte offset where particle records start in the file.

This is typically after any file header. Default implementation returns 0.

Returns

std::size_t The byte offset of the first particle record

Reimplemented in ParticleZoo::EGSphspFile::Reader, and ParticleZoo::IAEAphspFile::Reader.

6.12.3.26 getParticlesRead() [1/2]

```
std::uint64_t ParticleZoo::PhaseSpaceFileReader::getParticlesRead ( )  [inline], [virtual]
```

Get the number of particles that have been read so far.

This excludes metadata particles and skipped particles.

Returns

std::uint64_t The number of particles read

6.12.3.27 getParticlesRead() [2/2]

```
std::uint64_t ParticleZoo::PhaseSpaceFileReader::getParticlesRead (
            bool includeSkippedParticles )  [inline], [protected], [virtual]
```

Get the number of particles read with optional inclusion of skipped particles (including pseudo-particles).

Parameters

includeSkippedParticles	Whether to include pseudo-particles and particles skipped by moveToParticle()

Returns

std::uint64_t The number of particles read

6.12.3.28 getPHSPFormat()

```
const std::string ParticleZoo::PhaseSpaceFileReader::getPHSPFormat ( ) const  [inline]
```

Get the phase space file format identifier.

Returns

const std::string The format identifier (e.g., "IAEA", "EGS", "TOPAS")

6.12.3.29 getUserOptions()

```
const UserOptions & ParticleZoo::PhaseSpaceFileReader::getUserOptions ( ) const  [inline], [protected]
```

Get the user options that were passed to the constructor.

Returns

const UserOptions& Reference to the user options

6.12.3.30 hasMoreParticles()

```
bool ParticleZoo::PhaseSpaceFileReader::hasMoreParticles ( )  [virtual]
```

Check if there are more particles to read in the file.

Returns

true if there are more particles available to read
false if the end of file has been reached

6.12.3.31 isPxConstant()

```
bool ParticleZoo::PhaseSpaceFileReader::isPxConstant ( ) const  [inline]
```

Check if the X-component of momentum is constant for all particles.

Returns

true if Px is constant across all particles
false if Px varies between particles

6.12.3.32 isPyConstant()

```
bool ParticleZoo::PhaseSpaceFileReader::isPyConstant ( ) const  [inline]
```

Check if the Y-component of momentum is constant for all particles.

Returns

true if Py is constant across all particles
false if Py varies between particles

6.12.3.33 isPzConstant()

```
bool ParticleZoo::PhaseSpaceFileReader::isPzConstant ( ) const  [inline]
```

Check if the Z-component of momentum is constant for all particles.

Returns

true if Pz is constant across all particles
false if Pz varies between particles

6.12.3.34 isWeightConstant()

```
bool ParticleZoo::PhaseSpaceFileReader::isWeightConstant ( ) const  [inline]
```

Check if the statistical weight is constant for all particles.

Returns

true if weight is constant across all particles
false if weight varies between particles

6.12.3.35 isXConstant()

```
bool ParticleZoo::PhaseSpaceFileReader::isXConstant ( ) const  [inline]
```

Check if the X coordinate is constant for all particles.

Returns

true if X coordinate is constant across all particles
false if X coordinate varies between particles

6.12.3.36 isYConstant()

```
bool ParticleZoo::PhaseSpaceFileReader::isYConstant ( ) const  [inline]
```

Check if the Y coordinate is constant for all particles.

Returns

true if Y coordinate is constant across all particles

false if Y coordinate varies between particles

6.12.3.37 isZConstant()

```
bool ParticleZoo::PhaseSpaceFileReader::isZConstant ( ) const  [inline]
```

Check if the Z coordinate is constant for all particles.

Returns

true if Z coordinate is constant across all particles

false if Z coordinate varies between particles

6.12.3.38 moveToParticle()

```
void ParticleZoo::PhaseSpaceFileReader::moveToParticle (
            std::uint64_t particleIndex )
```

Move the file position to a specific particle index.

Allows random access to particles within the file. The next call to getNextParticle() will return the particle at the specified index.

Parameters

particleIndex	Zero-based index of the particle to move to

6.12.3.39 readASCIIParticle()

```
Particle ParticleZoo::PhaseSpaceFileReader::readASCIIParticle (
            const std::string & line )  [inline], [protected], [virtual]
```

Read a particle from ASCII data.

Must be implemented by derived classes that support ASCII format. The default implementation throws an exception.

Parameters

line	The ASCII line containing the particle data

Returns

Particle The particle object parsed from ASCII data

Exceptions

std::runtime_error	if not implemented for ASCII format

Reimplemented in ParticleZoo::penEasyphspFile::Reader, and ParticleZoo::TOPASphspFile::Reader.

6.12.3.40 readBinaryParticle()

```
Particle ParticleZoo::PhaseSpaceFileReader::readBinaryParticle (
            ByteBuffer & buffer )  [inline], [protected], [virtual]
```

Read a particle from binary data.

Must be implemented by derived classes that support binary format. The default implementation throws an exception.

Parameters

buffer	The byte buffer containing the particle data

Returns

Particle The particle object parsed from binary data

Exceptions

std::runtime_error	if not implemented for binary format

Reimplemented in ParticleZoo::EGSphspFile::Reader, ParticleZoo::IAEAphspFile::Reader, and ParticleZoo::TOPASphspFile::Reader.

6.12.3.41 readParticleManually()

```
Particle ParticleZoo::PhaseSpaceFileReader::readParticleManually ( )  [inline], [protected], [virtual]
```

Read a particle manually (for formats requiring third-party I/O).

Can be implemented by derived classes to support manual file I/O, circumventing the internal file stream and buffer.

Must be implemented by derived classes that specify FormatType::NONE. The default implementation throws an exception.

Returns

Particle The manually entered particle object

Exceptions

std::runtime_error	if not implemented

Reimplemented in ParticleZoo::ROOT::Reader.

6.12.3.42 setByteOrder()

```
void ParticleZoo::PhaseSpaceFileReader::setByteOrder (
             ByteOrder byteOrder )  [inline], [protected]
```

Set the byte order for binary data interpretation.

Parameters

byteOrder	The byte order to use (little-endian, big-endian, or PDP-endian)

6.12.3.43 setCommentMarkers()

```
void ParticleZoo::PhaseSpaceFileReader::setCommentMarkers (
            const std::vector< std::string > & commentMarkers )  [inline]
```

Set comment markers for ASCII format files.

Defines the strings that mark comment lines in ASCII format files. Lines beginning with these markers will be ignored during parsing.

Parameters

commentMarkers	Vector of strings that indicate comment lines

6.12.3.44 setConstantPx()

```
void ParticleZoo::PhaseSpaceFileReader::setConstantPx (
            float Px )  [inline], [protected]
```

Set a constant X-component of the direction unit vector for all particles.

Parameters

Px	The constant Px value to set

6.12.3.45 setConstantPy()

```
void ParticleZoo::PhaseSpaceFileReader::setConstantPy (
            float Py )  [inline], [protected]
```

Set a constant Y-component of the direction unit vector for all particles.

Parameters

Py	The constant Py value to set

6.12.3.46 setConstantPz()

```
void ParticleZoo::PhaseSpaceFileReader::setConstantPz (
             float Pz )  [inline], [protected]
```

Set a constant Z-component of the direction unit vector for all particles.

Parameters

Pz	The constant Pz value to set

6.12.3.47 setConstantWeight()

```
void ParticleZoo::PhaseSpaceFileReader::setConstantWeight (
             float weight )  [inline], [protected]
```

Set a constant statistical weight for all particles.

Parameters

weight	The constant weight value to set

6.12.3.48 setConstantX()

```
void ParticleZoo::PhaseSpaceFileReader::setConstantX (
             float X )  [inline], [protected]
```

Set a constant X coordinate value for all particles.

Parameters

X	The constant X coordinate value to set

6.12.3.49 setConstantY()

```
void ParticleZoo::PhaseSpaceFileReader::setConstantY (
             float Y )  [inline], [protected]
```

Set a constant Y coordinate value for all particles.

Parameters

Y	The constant Y coordinate value to set

6.12.3.50 setConstantZ()

```
void ParticleZoo::PhaseSpaceFileReader::setConstantZ (
             float Z )  [inline], [protected]
```

Set a constant Z coordinate value for all particles.

Parameters

Z	The constant Z coordinate value to set

The documentation for this class was generated from the following files:

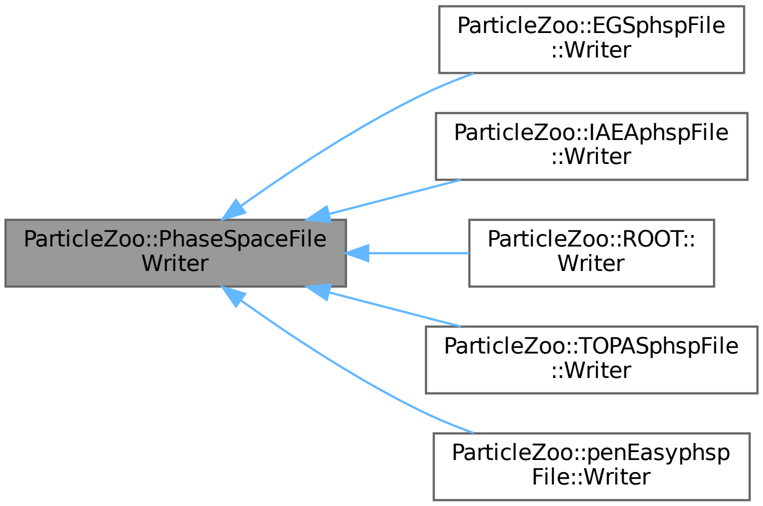
- include/particlezoo/PhaseSpaceFileReader.h
- src/PhaseSpaceFileReader.cc

6.13 ParticleZoo::PhaseSpaceFileWriter Class Reference

Base class for writing phase space files.

```
#include <particlezoo/PhaseSpaceFileWriter.h>
```

Inheritance diagram for ParticleZoo::PhaseSpaceFileWriter:


ParticleZoo::EGSphspFile ::Writer
ParticleZoo::IAEAphspFile ::Writer
ParticleZoo::PhaseSpaceFile Writer
ParticleZoo::ROOT:: Writer
ParticleZoo::TOPASphspFile ::Writer
ParticleZoo::penEasyphsp File::Writer

Public Member Functions

- PhaseSpaceFileWriter (const std::string &phspFormat, const std::string &fileName, const UserOptions &user↩ Options, FormatType formatType=FormatType::BINARY, const FixedValues fixedValues=FixedValues(), unsigned int bufferSize=DEFAULT_BUFFER_SIZE)

  Construct a new Phase Space File Writer object.
- virtual ∼PhaseSpaceFileWriter ()

  Destroy the Phase Space File Writer object.
- void addAdditionalHistories (std::uint64_t additionalHistories)

  Add additional Monte Carlo histories to the count.
- void close ()

  Close the phase space file and finalize writing.
- ByteOrder getByteOrder () const

Get the byte order used for binary data writing.
- float getConstantPx () const

  Get the constant X-component of the direction unit vector (if constant).
- float getConstantPy () const

  Get the constant Y-component of the direction unit vector (if constant).
- float getConstantPz () const

  Get the constant Z-component of the direction unit vector (if constant).
- float getConstantWeight () const

  Get the constant statistical weight value (if constant).
- float getConstantX () const

  Get the constant X coordinate value (if constant).
- float getConstantY () const

  Get the constant Y coordinate value (if constant).
- float getConstantZ () const

  Get the constant Z coordinate value (if constant).
- const std::string getFileName () const

  Get the filename where the phase space file is being written.
- const FixedValues getFixedValues () const

  Get the fixed values configuration.
- virtual std::uint64_t getHistoriesWritten () const

  Get the number of Monte Carlo histories that have been written.
- virtual std::uint64_t getMaximumSupportedParticles () const =0

  Get the maximum number of particles this writer can support.
- std::uint64_t getParticlesWritten () const

  Get the number of particles that have been written to the file.
- const std::string getPHSPFormat () const

  Get the phase space file format identifier.
- bool isPxConstant () const

  Check if the X-component of the direction unit vector is set to a constant value for all particles.
- bool isPyConstant () const

  Check if the Y-component of the direction unit vector is set to a constant value for all particles.
- bool isPzConstant () const

  Check if the Z-component of the direction unit vector is set to a constant value for all particles.
- bool isWeightConstant () const

  Check if the statistical weight is set to a constant value for all particles.
- bool isXConstant () const

  Check if the X coordinate is set to a constant value for all particles.
- bool isYConstant () const

  Check if the Y coordinate is set to a constant value for all particles.
- bool isZConstant () const

  Check if the Z coordinate is set to a constant value for all particles.
- virtual void writeParticle (Particle particle)

  Write a particle to the phase space file.

Static Public Member Functions

- static std::vector< CLICommand > getCLICommands ()

  Get command line interface commands supported by this writer.

Protected Member Functions

- virtual bool accountForAdditionalHistories (std::uint64_t additionalHistories)

  Handle accounting for simulation histories that produced no particles.
- virtual bool canHaveConstantPx () const

  Check if this format supports constant X-component of the direction unit vector.
- virtual bool canHaveConstantPy () const

  Check if this format supports constant Y-component of the direction unit vector.
- virtual bool canHaveConstantPz () const

  Check if this format supports constant Z-component of the direction unit vector.
- virtual bool canHaveConstantWeight () const

  Check if this format supports constant statistical weights.
- virtual bool canHaveConstantX () const

  Check if this format supports constant X coordinates.
- virtual bool canHaveConstantY () const

  Check if this format supports constant Y coordinates.
- virtual bool canHaveConstantZ () const

  Check if this format supports constant Z coordinates.
- virtual bool canWritePseudoParticlesExplicitly () const

  Check if this format can write pseudo-particles explicitly.
- virtual void fixedValuesHaveChanged ()

  Called when fixed values have been changed.
- virtual size_t getMaximumASCIILineLength () const

  Get the maximum line length for ASCII format files.
- virtual std::size_t getParticleRecordLength () const

  Get the length in bytes of each particle record.
- virtual std::size_t getParticleRecordStartOffset () const

  Get the byte offset where particle records start in the file.
- virtual std::uint64_t getPendingHistories () const

  Get the number of pending histories to account for.
- const UserOptions & getUserOptions () const

  Get the user options that were passed to the constructor.
- void setByteOrder (ByteOrder byteOrder)

  Set the byte order for binary data writing.
- void setConstantPx (float Px)

  Set a constant X-component of the direction unit vector for all particles.

- void setConstantPy (float Py)

    Set a constant Y-component of the direction unit vector for all particles.
- void setConstantPz (float Pz)

    Set a constant Z-component of the direction unit vector for all particles.
- void setConstantWeight (float weight)

    Set a constant statistical weight for all particles.
- void setConstantX (float X)

    Set a constant X coordinate value for all particles.
- void setConstantY (float Y)

    Set a constant Y coordinate value for all particles.
- void setConstantZ (float Z)

    Set a constant Z coordinate value for all particles.
- virtual const std::string writeASCIIParticle (Particle &particle)

    Write a particle in ASCII format as a string.
- virtual void writeBinaryParticle (ByteBuffer &buffer, Particle &particle)

    Write a particle in binary format to a byte buffer.
- virtual void writeHeaderData (ByteBuffer &buffer)=0

    Write header data to a byte buffer.
- virtual void writeParticleManually (Particle &particle)

    Write a particle manually (for formats requiring third-party I/O).

6.13.1 Detailed Description

Base class for writing phase space files.

This abstract class provides a unified interface for writing particle phase space files to different simulation formats (EGS, IAEA, TOPAS, etc.). It handles both binary and ASCII file formats, provides buffering for efficient I/O, and supports statistics tracking and format-specific optimizations. In cases where I/O must be handled by a third-party library (e.g., ROOT), this class also provides a framework for manually writing particles.

6.13.2 Constructor & Destructor Documentation

6.13.2.1 PhaseSpaceFileWriter()

```
ParticleZoo::PhaseSpaceFileWriter::PhaseSpaceFileWriter (
             const std::string & phspFormat,
             const std::string & fileName,
             const UserOptions & userOptions,
             FormatType formatType = FormatType::BINARY,
             const FixedValues fixedValues = FixedValues(),
             unsigned int bufferSize = DEFAULT_BUFFER_SIZE )
```

Construct a new Phase Space File Writer object.

Parameters

phspFormat	The format identifier of the phase space file (e.g., "IAEA", "EGS", "TOPAS")
fileName	The path where the phase space file will be written
userOptions	User-defined options for writing behavior
formatType	The format type (BINARY or ASCII), defaults to BINARY
fixedValues	Pre-defined constant values for certain particle properties
bufferSize	Size of the internal buffer for writing, defaults to DEFAULT_BUFFER_SIZE

6.13.2.2 ∼PhaseSpaceFileWriter()

```
ParticleZoo::PhaseSpaceFileWriter::∼PhaseSpaceFileWriter ( )  [virtual]
```

Destroy the Phase Space File Writer object.

Ensures proper cleanup by closing the file and flushing any remaining buffered data.

6.13.3 Member Function Documentation

6.13.3.1 accountForAdditionalHistories()

```
bool ParticleZoo::PhaseSpaceFileWriter::accountForAdditionalHistories (
             std::uint64_t additionalHistories )  [inline], [protected], [virtual]
```

Handle accounting for simulation histories that produced no particles.

Called by addAdditionalHistories() to handle format-specific requirements for empty histories. Some formats need special handling such as writing pseudo-particles or updating header counters.

The default implementation returns true, indicating that the base class should automatically increment the history counter. Derived classes can override this to handle it manually (e.g., by writing additional pseudo-particles)

Parameters

additionalHistories	The number of additional (empty) histories

Returns

true if the base class should automatically increment the history counter

false if the derived class handles it manually (e.g., by writing additional pseudo-particles)

Reimplemented in ParticleZoo::TOPASphspFile::Writer.

6.13.3.2 addAdditionalHistories()

```
void ParticleZoo::PhaseSpaceFileWriter::addAdditionalHistories (
             std::uint64_t additionalHistories )  [inline]
```

Add additional Monte Carlo histories to the count.

Used to account for simulation histories that produced no particles to write. Some formats may need special handling for empty histories.

Parameters

additionalHistories	The number of additional (empty) histories to account for

6.13.3.3 canHaveConstantPx()

```
bool ParticleZoo::PhaseSpaceFileWriter::canHaveConstantPx ( ) const  [inline], [protected], [virtual]
```

Check if this format supports constant X-component of the direction unit vector.

Derived classes can override this to indicate format-specific capabilities. Default implementation returns false.

Returns

true if constant Px is supported by this format

false if constant Px is not supported

Reimplemented in ParticleZoo::IAEAphspFile::Writer.

6.13.3.4 canHaveConstantPy()

`bool ParticleZoo::PhaseSpaceFileWriter::canHaveConstantPy ( ) const [inline], [protected], [virtual]`

Check if this format supports constant Y-component of the direction unit vector.

Derived classes can override this to indicate format-specific capabilities. Default implementation returns false.

Returns

true if constant Py is supported by this format

false if constant Py is not supported

Reimplemented in ParticleZoo::IAEAphspFile::Writer.

6.13.3.5 canHaveConstantPz()

`bool ParticleZoo::PhaseSpaceFileWriter::canHaveConstantPz ( ) const [inline], [protected], [virtual]`

Check if this format supports constant Z-component of the direction unit vector.

Derived classes can override this to indicate format-specific capabilities. Default implementation returns false.

Returns

true if constant Pz is supported by this format

false if constant Pz is not supported

Reimplemented in ParticleZoo::IAEAphspFile::Writer.

6.13.3.6 canHaveConstantWeight()

`bool ParticleZoo::PhaseSpaceFileWriter::canHaveConstantWeight ( ) const [inline], [protected], [virtual]`

Check if this format supports constant statistical weights.

Derived classes can override this to indicate format-specific capabilities. Default implementation returns false.

Returns

true if constant weights are supported by this format

false if constant weights are not supported

Reimplemented in ParticleZoo::IAEAphspFile::Writer.

6.13.3.7 canHaveConstantX()

`bool ParticleZoo::PhaseSpaceFileWriter::canHaveConstantX ( ) const [inline], [protected], [virtual]`

Check if this format supports constant X coordinates.

Derived classes can override this to indicate format-specific capabilities. Default implementation returns false.

Returns

true if constant X coordinates are supported by this format

false if constant X coordinates are not supported

Reimplemented in ParticleZoo::IAEAphspFile::Writer.

6.13.3.8 canHaveConstantY()

`bool ParticleZoo::PhaseSpaceFileWriter::canHaveConstantY ( ) const [inline], [protected], [virtual]`

Check if this format supports constant Y coordinates.

Derived classes can override this to indicate format-specific capabilities. Default implementation returns false.

Returns

true if constant Y coordinates are supported by this format

false if constant Y coordinates are not supported

Reimplemented in ParticleZoo::IAEAphspFile::Writer.

6.13.3.9 canHaveConstantZ()

`bool ParticleZoo::PhaseSpaceFileWriter::canHaveConstantZ ( ) const [inline], [protected], [virtual]`

Check if this format supports constant Z coordinates.

Derived classes can override this to indicate format-specific capabilities. Default implementation returns false.

Returns

true if constant Z coordinates are supported by this format

false if constant Z coordinates are not supported

Reimplemented in ParticleZoo::IAEAphspFile::Writer.

6.13.3.10 canWritePseudoParticlesExplicitly()

```
bool ParticleZoo::PhaseSpaceFileWriter::canWritePseudoParticlesExplicitly ( ) const  [inline],
[protected], [virtual]
```

Check if this format can write pseudo-particles explicitly.

Derived classes can override this to indicate if they support writing pseudo-particles (metadata particles) explicitly to the file. Default implementation returns false.

Returns

true if pseudo-particles can be written explicitly

false if explicit pseudo-particle writing is not supported

Reimplemented in ParticleZoo::TOPASphspFile::Writer.

6.13.3.11 close()

```
void ParticleZoo::PhaseSpaceFileWriter::close ( )
```

Close the phase space file and finalize writing.

Flushes any remaining buffered data, writes the file header, and closes the file handle. The writer cannot be used after calling this method.

6.13.3.12 fixedValuesHaveChanged()

```
virtual void ParticleZoo::PhaseSpaceFileWriter::fixedValuesHaveChanged ( )  [inline], [protected],
[virtual]
```

Called when fixed values have been changed.

Derived classes can override this to perform any necessary updates when constant values are modified. Default implementation does nothing.

Reimplemented in ParticleZoo::IAEAphspFile::Writer.

6.13.3.13 getByteOrder()

```
ByteOrder ParticleZoo::PhaseSpaceFileWriter::getByteOrder ( ) const  [inline]
```

Get the byte order used for binary data writing.

Returns

ByteOrder The current byte order (little-endian, big-endian, or PDP-endian)

6.13.3.14 getCLICommands()

```
std::vector< CLICommand > ParticleZoo::PhaseSpaceFileWriter::getCLICommands ( )  [static]
```

Get command line interface commands supported by this writer.

Returns a vector of CLI commands that can be used with this writer type.

Returns

std::vector<CLICommand> Vector of supported CLI commands

6.13.3.15 getConstantPx()

```
float ParticleZoo::PhaseSpaceFileWriter::getConstantPx ( ) const  [inline]
```

Get the constant X-component of the direction unit vector (if constant).

Returns

float The constant Px value

Exceptions

std::runtime_error	if Px is not set to constant

6.13.3.16 getConstantPy()

```
float ParticleZoo::PhaseSpaceFileWriter::getConstantPy ( ) const  [inline]
```

Get the constant Y-component of the direction unit vector (if constant).

Returns

float The constant Py value

Exceptions

std::runtime_error	if Py is not set to constant

6.13.3.17 getConstantPz()

```
float ParticleZoo::PhaseSpaceFileWriter::getConstantPz ( ) const  [inline]
```

Get the constant Z-component of the direction unit vector (if constant).

Returns

float The constant Pz value

Exceptions

std::runtime_error	if Pz is not set to constant

6.13.3.18 getConstantWeight()

```
float ParticleZoo::PhaseSpaceFileWriter::getConstantWeight ( ) const  [inline]
```

Get the constant statistical weight value (if constant).

Returns

float The constant weight value

Exceptions

std::runtime_error	if weight is not set to constant

6.13.3.19 getConstantX()

```
float ParticleZoo::PhaseSpaceFileWriter::getConstantX ( ) const  [inline]
```

Get the constant X coordinate value (if constant).

Returns

float The constant X coordinate value

Exceptions

std::runtime_error	if X is not set to constant

6.13.3.20 getConstantY()

```
float ParticleZoo::PhaseSpaceFileWriter::getConstantY ( ) const  [inline]
```

Get the constant Y coordinate value (if constant).

Returns

float The constant Y coordinate value

Exceptions

std::runtime_error	if Y is not set to constant

6.13.3.21 getConstantZ()

```
float ParticleZoo::PhaseSpaceFileWriter::getConstantZ ( ) const  [inline]
```

Get the constant Z coordinate value (if constant).

Returns

float The constant Z coordinate value

Exceptions

std::runtime_error	if Z is not set to constant

6.13.3.22 getFileName()

```
const std::string ParticleZoo::PhaseSpaceFileWriter::getFileName ( ) const  [inline]
```

Get the filename where the phase space file is being written.

Returns

const std::string The filename/path of the output file

6.13.3.23 getFixedValues()

```
const FixedValues ParticleZoo::PhaseSpaceFileWriter::getFixedValues ( ) const  [inline]
```

Get the fixed values configuration.

Returns

const FixedValues The complete fixed values structure

6.13.3.24 getHistoriesWritten()

```
std::uint64_t ParticleZoo::PhaseSpaceFileWriter::getHistoriesWritten ( ) const  [inline], [virtual]
```

Get the number of Monte Carlo histories that have been written.

Returns

std::uint64_t The number of histories written to the file

6.13.3.25 getMaximumASCIILineLength()

```
size_t ParticleZoo::PhaseSpaceFileWriter::getMaximumASCIILineLength ( ) const  [inline], [protected],
[virtual]
```

Get the maximum line length for ASCII format files.

Must be implemented by derived classes that support ASCII format. Used for buffer allocation and writing optimization.

Returns

std::size_t The maximum length of ASCII lines in characters

Exceptions

std::runtime_error	if not implemented for ASCII format

Reimplemented in ParticleZoo::penEasyphspFile::Writer, and ParticleZoo::TOPASphspFile::Writer.

6.13.3.26 getMaximumSupportedParticles()

```
virtual std::uint64_t ParticleZoo::PhaseSpaceFileWriter::getMaximumSupportedParticles ( ) const
[pure virtual]
```

Get the maximum number of particles this writer can support.

This is a pure virtual method that must be implemented by derived classes as the maximum can vary by format.

Returns

std::uint64_t The maximum number of particles supported

Implemented in ParticleZoo::EGSphspFile::Writer, ParticleZoo::IAEAphspFile::Writer, ParticleZoo::penEasyphspFile::Writer, ParticleZoo::ROOT::Writer, and ParticleZoo::TOPASphspFile::Writer.

6.13.3.27 getParticleRecordLength()

```
std::size_t ParticleZoo::PhaseSpaceFileWriter::getParticleRecordLength ( ) const  [inline], [protected],
[virtual]
```

Get the length in bytes of each particle record.

Must be implemented by derived classes for binary formatted files.

Returns

std::size_t The length of each particle record in bytes

Exceptions

std::runtime_error	if not implemented for binary format

Reimplemented in ParticleZoo::EGSphspFile::Writer, ParticleZoo::IAEAphspFile::Writer, and ParticleZoo::TOPASphspFile::Writer.

6.13.3.28 getParticleRecordStartOffset()

```
std::size_t ParticleZoo::PhaseSpaceFileWriter::getParticleRecordStartOffset ( ) const  [inline],
[protected], [virtual]
```

Get the byte offset where particle records start in the file.

This is typically after any file header. Default implementation returns 0.

Returns

std::size_t The byte offset of the first particle record

Reimplemented in ParticleZoo::EGSphspFile::Writer, and ParticleZoo::penEasyphspFile::Writer.

6.13.3.29 getParticlesWritten()

```
std::uint64_t ParticleZoo::PhaseSpaceFileWriter::getParticlesWritten ( ) const  [inline]
```

Get the number of particles that have been written to the file.

Returns

std::uint64_t The number of particles written (excludes pseudo-particles)

6.13.3.30 getPendingHistories()

```
std::uint64_t ParticleZoo::PhaseSpaceFileWriter::getPendingHistories ( ) const  [inline], [protected],
[virtual]
```

Get the number of pending histories to account for.

Used internally to track histories that have not yet been written to the file (e.g., empty histories).

Override in derived classes if special handling is needed.

Returns

std::uint64_t The number of pending histories

Reimplemented in ParticleZoo::TOPASphspFile::Writer.

6.13.3.31 getPHSPFormat()

```
const std::string ParticleZoo::PhaseSpaceFileWriter::getPHSPFormat ( ) const  [inline]
```

Get the phase space file format identifier.

Returns

const std::string The format identifier (e.g., "IAEA", "EGS", "TOPAS")

6.13.3.32 getUserOptions()

```
const UserOptions & ParticleZoo::PhaseSpaceFileWriter::getUserOptions ( ) const  [inline], [protected]
```

Get the user options that were passed to the constructor.

Returns

const UserOptions& Reference to the user options

6.13.3.33 isPxConstant()

```
bool ParticleZoo::PhaseSpaceFileWriter::isPxConstant ( ) const  [inline]
```

Check if the X-component of the direction unit vector is set to a constant value for all particles.

Returns

true if Px is constant across all particles
false if Px varies between particles

6.13.3.34 isPyConstant()

```
bool ParticleZoo::PhaseSpaceFileWriter::isPyConstant ( ) const  [inline]
```

Check if the Y-component of the direction unit vector is set to a constant value for all particles.

Returns

true if Py is constant across all particles
false if Py varies between particles

6.13.3.35 isPzConstant()

```
bool ParticleZoo::PhaseSpaceFileWriter::isPzConstant ( ) const  [inline]
```

Check if the Z-component of the direction unit vector is set to a constant value for all particles.

Returns

true if Pz is constant across all particles
false if Pz varies between particles

6.13.3.36 isWeightConstant()

```
bool ParticleZoo::PhaseSpaceFileWriter::isWeightConstant ( ) const  [inline]
```

Check if the statistical weight is set to a constant value for all particles.

Returns

true if weight is constant across all particles
false if weight varies between particles

6.13.3.37 isXConstant()

```
bool ParticleZoo::PhaseSpaceFileWriter::isXConstant ( ) const  [inline]
```

Check if the X coordinate is set to a constant value for all particles.

Returns

true if X coordinate is constant across all particles
false if X coordinate varies between particles

6.13.3.38 isYConstant()

```
bool ParticleZoo::PhaseSpaceFileWriter::isYConstant ( ) const  [inline]
```

Check if the Y coordinate is set to a constant value for all particles.

Returns

true if Y coordinate is constant across all particles
false if Y coordinate varies between particles

6.13.3.39 isZConstant()

```
bool ParticleZoo::PhaseSpaceFileWriter::isZConstant ( ) const  [inline]
```

Check if the Z coordinate is set to a constant value for all particles.

Returns

true if Z coordinate is constant across all particles

false if Z coordinate varies between particles

6.13.3.40 setByteOrder()

```
void ParticleZoo::PhaseSpaceFileWriter::setByteOrder (
             ByteOrder byteOrder )  [inline], [protected]
```

Set the byte order for binary data writing.

Parameters

byteOrder	The byte order to use (little-endian, big-endian, or PDP-endian)

6.13.3.41 setConstantPx()

```
void ParticleZoo::PhaseSpaceFileWriter::setConstantPx (
             float Px )  [inline], [protected]
```

Set a constant X-component of the direction unit vector for all particles.

Parameters

Px	The constant Px value to set

6.13.3.42 setConstantPy()

```
void ParticleZoo::PhaseSpaceFileWriter::setConstantPy (
             float Py )  [inline], [protected]
```

Set a constant Y-component of the direction unit vector for all particles.

Parameters

Py	The constant Py value to set

6.13.3.43 setConstantPz()

```
void ParticleZoo::PhaseSpaceFileWriter::setConstantPz (
             float Pz )  [inline], [protected]
```

Set a constant Z-component of the direction unit vector for all particles.

Parameters

Pz	The constant Pz value to set

6.13.3.44 setConstantWeight()

```
void ParticleZoo::PhaseSpaceFileWriter::setConstantWeight (
             float weight )  [inline], [protected]
```

Set a constant statistical weight for all particles.

Parameters

weight	The constant weight value to set

6.13.3.45 setConstantX()

```
void ParticleZoo::PhaseSpaceFileWriter::setConstantX (
             float X )  [inline], [protected]
```

Set a constant X coordinate value for all particles.

Parameters

X	The constant X coordinate value to set

6.13.3.46 setConstantY()

```
void ParticleZoo::PhaseSpaceFileWriter::setConstantY (
            float Y )  [inline], [protected]
```

Set a constant Y coordinate value for all particles.

Parameters

Y	The constant Y coordinate value to set

6.13.3.47 setConstantZ()

```
void ParticleZoo::PhaseSpaceFileWriter::setConstantZ (
            float Z )  [inline], [protected]
```

Set a constant Z coordinate value for all particles.

Parameters

Z	The constant Z coordinate value to set

6.13.3.48 writeASCIIParticle()

```
const std::string ParticleZoo::PhaseSpaceFileWriter::writeASCIIParticle (
            Particle & particle )  [inline], [protected], [virtual]
```

Write a particle in ASCII format as a string.

Must be implemented by derived classes that support ASCII format. The default implementation throws an exception.

Parameters

particle	The particle object to write

Returns

const std::string The ASCII representation of the particle

Exceptions

std::runtime_error	if not implemented for ASCII format

Reimplemented in ParticleZoo::penEasyphspFile::Writer, and ParticleZoo::TOPASphspFile::Writer.

6.13.3.49 writeBinaryParticle()

```
void ParticleZoo::PhaseSpaceFileWriter::writeBinaryParticle (
            ByteBuffer & buffer,
            Particle & particle )  [inline], [protected], [virtual]
```

Write a particle in binary format to a byte buffer.

Must be implemented by derived classes that support binary format. The default implementation throws an exception.

Parameters

buffer	The byte buffer to write particle data into
particle	The particle object to write

Exceptions

std::runtime_error	if not implemented for binary format

Reimplemented in ParticleZoo::EGSphspFile::Writer, ParticleZoo::IAEAphspFile::Writer, and ParticleZoo::TOPASphspFile::Writer.

6.13.3.50 writeHeaderData()

```
virtual void ParticleZoo::PhaseSpaceFileWriter::writeHeaderData (
            ByteBuffer & buffer )  [protected], [pure virtual]
```

Write header data to a byte buffer.

This is a pure virtual method that must be implemented by derived classes to write format-specific header information.

Parameters

buffer	The byte buffer to write header data into

Implemented in ParticleZoo::EGSphspFile::Writer, ParticleZoo::IAEAphspFile::Writer, ParticleZoo::penEasyphspFile::Writer, ParticleZoo::ROOT::Writer, and ParticleZoo::TOPASphspFile::Writer.

6.13.3.51 writeParticle()

```
void ParticleZoo::PhaseSpaceFileWriter::writeParticle (
            Particle particle )  [virtual]
```

Write a particle to the phase space file.

Writes the given particle to the file using the appropriate format. For binary or ASCII files, the particle is automatically buffered and written when the buffer is full. Applies any constant values that have been set before writing.

Parameters

particle	The particle object to write to the file

6.13.3.52 writeParticleManually()

```
void ParticleZoo::PhaseSpaceFileWriter::writeParticleManually (
            Particle & particle )  [inline], [protected], [virtual]
```

Write a particle manually (for formats requiring third-party I/O).

Can be implemented by derived classes to support manual file I/O, circumventing the internal file stream and buffer.

Must be implemented by derived classes that specify FormatType::NONE. The default implementation throws an exception.

Parameters

particle	The particle object to write manually

Exceptions

std::runtime_error	if not implemented

Reimplemented in ParticleZoo::ROOT::Writer.

The documentation for this class was generated from the following files:

- include/particlezoo/PhaseSpaceFileWriter.h
- src/PhaseSpaceFileWriter.cc

6.14 ParticleZoo::ROOT::BranchInfo Struct Reference

Configuration for ROOT TTree branch mapping.

```
#include <particlezoo/ROOT/ROOTphsp.h>
```

Public Attributes

- std::string **branchName**

  Name of the ROOT TTree branch.
- double **unitFactor**

  Unit conversion factor to internal units.

6.14.1 Detailed Description

Configuration for ROOT TTree branch mapping.

Contains the branch name and unit conversion factor for mapping physical quantities to ROOT TTree branches.

The documentation for this struct was generated from the following file:

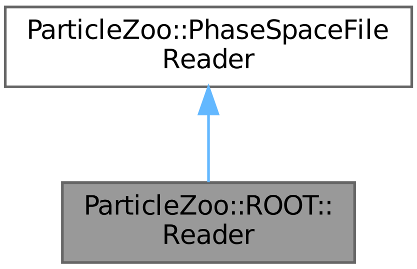
- include/particlezoo/ROOT/ROOTphsp.h

6.15 ParticleZoo::ROOT::Reader Class Reference

ROOT format phase space file reader.

```
#include <particlezoo/ROOT/ROOTphsp.h>
```

Inheritance diagram for ParticleZoo::ROOT::Reader:


ParticleZoo::PhaseSpaceFile Reader
ParticleZoo::ROOT:: Reader

Public Member Functions

- Reader (const std::string &fileName, const UserOptions &options=UserOptions{})

  Construct a ROOT file reader with user options.
- ∼**Reader** () override

  Destructor - closes ROOT file and cleans up resources.
- std::uint64_t getNumberOfOriginalHistories () const override

  Get number of original Monte Carlo histories.
- std::uint64_t getNumberOfParticles () const override

  Get total number of particles in the ROOT file.

Public Member Functions inherited from ParticleZoo::PhaseSpaceFileReader

- PhaseSpaceFileReader (const std::string &phspFormat, const std::string &fileName, const UserOptions &user↩ Options, FormatType formatType=FormatType::BINARY, const FixedValues fixedValues=FixedValues(), unsigned int bufferSize=DEFAULT_BUFFER_SIZE)

  Construct a new Phase Space File Reader object.
- virtual ∼PhaseSpaceFileReader ()

Destroy the Phase Space File Reader object.

- void close ()

  Close the phase space file and clean up resources.
- float getConstantPx () const

  Get the constant X-component of the direction unit vector (if constant).
- float getConstantPy () const

  Get the constant Y-component of the direction unit vector (if constant).
- float getConstantPz () const

  Get the constant Z-component of the direction unit vector (if constant).
- float getConstantWeight () const

  Get the constant statistical weight value (if constant).
- float getConstantX () const

  Get the constant X coordinate value (if constant).
- float getConstantY () const

  Get the constant Y coordinate value (if constant).
- float getConstantZ () const

  Get the constant Z coordinate value (if constant).
- const std::string getFileName () const

  Get the filename of the phase space file being read.
- std::uint64_t getFileSize () const

  Get the size of the phase space file in bytes.
- const FixedValues getFixedValues () const

  Get the fixed values configuration.
- virtual std::uint64_t getHistoriesRead ()

  Get the number of Monte Carlo histories that have been read so far. If the end of the file has been reached, this will return the total number of original histories unless more histories than expected have already been read - in which case it returns the actual count.
- Particle getNextParticle ()

  Get the next particle from the phase space file.
- virtual std::uint64_t getParticlesRead ()

  Get the number of particles that have been read so far.
- const std::string getPHSPFormat () const

  Get the phase space file format identifier.
- virtual bool hasMoreParticles ()

  Check if there are more particles to read in the file.
- bool isPxConstant () const

  Check if the X-component of momentum is constant for all particles.
- bool isPyConstant () const

  Check if the Y-component of momentum is constant for all particles.
- bool isPzConstant () const

  Check if the Z-component of momentum is constant for all particles.
- bool isWeightConstant () const

Check if the statistical weight is constant for all particles.

- bool isXConstant () const

  Check if the X coordinate is constant for all particles.
- bool isYConstant () const

  Check if the Y coordinate is constant for all particles.
- bool isZConstant () const

  Check if the Z coordinate is constant for all particles.
- void moveToParticle (std::uint64_t particleIndex)

  Move the file position to a specific particle index.
- void setCommentMarkers (const std::vector< std::string > &commentMarkers)

  Set comment markers for ASCII format files.

Static Public Member Functions

- static std::vector< CLICommand > getFormatSpecificCLICommands ()

  Get format-specific CLI commands for ROOT configuration.

Static Public Member Functions inherited from ParticleZoo::PhaseSpaceFileReader

- static std::vector< CLICommand > getCLICommands ()

  Get command line interface commands supported by this reader.

Protected Member Functions

- Particle readParticleManually () override

  Read next particle from ROOT TTree.

Protected Member Functions inherited from ParticleZoo::PhaseSpaceFileReader

- double calcThirdUnitComponent (double &u, double &v) const

  Calculate the third component of a unit vector from two components (double precision).
- float calcThirdUnitComponent (float &u, float &v) const

  Calculate the third component of a unit vector from two components (float precision).
- const ByteBuffer getHeaderData ()

  Get the file header data as a byte buffer.
- const ByteBuffer getHeaderData (std::size_t headerSize)

  Get a specific amount of header data as a byte buffer.
- virtual std::size_t getMaximumASCIILineLength () const

  Get the maximum line length for ASCII format files.

- Particle getNextParticle (bool countParticleInStatistics)

  Get the next particle with optional statistics counting control.
- std::size_t getNumberOfEntriesInFile () const

  Get the number of particle records that fit in the file.
- virtual std::size_t getParticleRecordLength () const

  Get the length in bytes of each particle record.
- virtual std::size_t getParticleRecordStartOffset () const

  Get the byte offset where particle records start in the file.
- virtual std::uint64_t getParticlesRead (bool includeSkippedParticles)

  Get the number of particles read with optional inclusion of skipped particles (including pseudo-particles).
- const UserOptions & getUserOptions () const

  Get the user options that were passed to the constructor.
- virtual Particle readASCIIParticle (const std::string &line)

  Read a particle from ASCII data.
- virtual Particle readBinaryParticle (ByteBuffer &buffer)

  Read a particle from binary data.
- void setByteOrder (ByteOrder byteOrder)

  Set the byte order for binary data interpretation.
- void setConstantPx (float Px)

  Set a constant X-component of the direction unit vector for all particles.
- void setConstantPy (float Py)

  Set a constant Y-component of the direction unit vector for all particles.
- void setConstantPz (float Pz)

  Set a constant Z-component of the direction unit vector for all particles.
- void setConstantWeight (float weight)

  Set a constant statistical weight for all particles.
- void setConstantX (float X)

  Set a constant X coordinate value for all particles.
- void setConstantY (float Y)

  Set a constant Y coordinate value for all particles.
- void setConstantZ (float Z)

  Set a constant Z coordinate value for all particles.

6.15.1 Detailed Description

ROOT format phase space file reader.

Reads particle data from ROOT TTree structures with configurable branch mappings. Supports multiple format presets (TOPAS, OpenGATE) and custom branch configurations.

6.15.2 Constructor & Destructor Documentation

6.15.2.1 Reader()

```
ParticleZoo::ROOT::Reader::Reader (
            const std::string & fileName,
            const UserOptions & options = UserOptions{} )
```

Construct a ROOT file reader with user options.

Parameters

fileName	Path to the ROOT file
options	User configuration options including branch mappings

Exceptions

std::runtime_error	If file cannot be opened or required branches missing

6.15.3 Member Function Documentation

6.15.3.1 getFormatSpecificCLICommands()

```
std::vector< CLICommand > ParticleZoo::ROOT::Reader::getFormatSpecificCLICommands ( )  [static]
```

Get format-specific CLI commands for ROOT configuration.

Returns

Vector of ROOT-specific CLI commands

6.15.3.2 getNumberOfOriginalHistories()

```
std::uint64_t ParticleZoo::ROOT::Reader::getNumberOfOriginalHistories ( ) const  [inline], [override],
[virtual]
```

Get number of original Monte Carlo histories.

Returns

Number of original histories

Implements ParticleZoo::PhaseSpaceFileReader.

6.15.3.3 getNumberOfParticles()

```
std::uint64_t ParticleZoo::ROOT::Reader::getNumberOfParticles ( ) const  [inline], [override],
[virtual]
```

Get total number of particles in the ROOT file.

Returns

Number of particles (TTree entries)

Implements ParticleZoo::PhaseSpaceFileReader.

6.15.3.4 readParticleManually()

```
Particle ParticleZoo::ROOT::Reader::readParticleManually ( )  [override], [protected], [virtual]
```

Read next particle from ROOT TTree.

Returns

Particle object with data from current TTree entry

Exceptions

std::runtime_error	If end of file reached or read error

Reimplemented from ParticleZoo::PhaseSpaceFileReader.

The documentation for this class was generated from the following files:

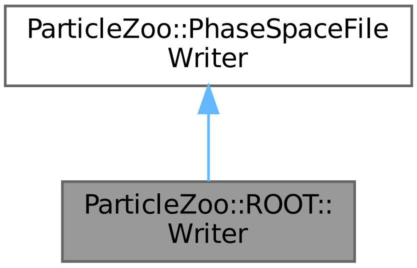
- include/particlezoo/ROOT/ROOTphsp.h
- src/ROOT/ROOTphsp.cc

6.16 ParticleZoo::ROOT::Writer Class Reference

ROOT format phase space file writer.

```
#include <particlezoo/ROOT/ROOTphsp.h>
```

Inheritance diagram for ParticleZoo::ROOT::Writer:


ParticleZoo::PhaseSpaceFile Writer
ParticleZoo::ROOT:: Writer

Public Member Functions

- Writer (const std::string &fileName, const UserOptions &options=UserOptions{})

  Construct a ROOT file writer with user options.
- ∼**Writer** () override

  Destructor - writes TTree and closes ROOT file.
- std::uint64_t getMaximumSupportedParticles () const override

  Get maximum number of particles that can be stored.

Public Member Functions inherited from ParticleZoo::PhaseSpaceFileWriter

- PhaseSpaceFileWriter (const std::string &phspFormat, const std::string &fileName, const UserOptions &user↩ Options, FormatType formatType=FormatType::BINARY, const FixedValues fixedValues=FixedValues(), unsigned int bufferSize=DEFAULT_BUFFER_SIZE)

  Construct a new Phase Space File Writer object.
- virtual ∼PhaseSpaceFileWriter ()

  Destroy the Phase Space File Writer object.
- void addAdditionalHistories (std::uint64_t additionalHistories)

Add additional Monte Carlo histories to the count.
- void close ()

  Close the phase space file and finalize writing.
- ByteOrder getByteOrder () const

  Get the byte order used for binary data writing.
- float getConstantPx () const

  Get the constant X-component of the direction unit vector (if constant).
- float getConstantPy () const

  Get the constant Y-component of the direction unit vector (if constant).
- float getConstantPz () const

  Get the constant Z-component of the direction unit vector (if constant).
- float getConstantWeight () const

  Get the constant statistical weight value (if constant).
- float getConstantX () const

  Get the constant X coordinate value (if constant).
- float getConstantY () const

  Get the constant Y coordinate value (if constant).
- float getConstantZ () const

  Get the constant Z coordinate value (if constant).
- const std::string getFileName () const

  Get the filename where the phase space file is being written.
- const FixedValues getFixedValues () const

  Get the fixed values configuration.
- virtual std::uint64_t getHistoriesWritten () const

  Get the number of Monte Carlo histories that have been written.
- std::uint64_t getParticlesWritten () const

  Get the number of particles that have been written to the file.
- const std::string getPHSPFormat () const

  Get the phase space file format identifier.
- bool isPxConstant () const

  Check if the X-component of the direction unit vector is set to a constant value for all particles.
- bool isPyConstant () const

  Check if the Y-component of the direction unit vector is set to a constant value for all particles.
- bool isPzConstant () const

  Check if the Z-component of the direction unit vector is set to a constant value for all particles.
- bool isWeightConstant () const

  Check if the statistical weight is set to a constant value for all particles.
- bool isXConstant () const

  Check if the X coordinate is set to a constant value for all particles.
- bool isYConstant () const

  Check if the Y coordinate is set to a constant value for all particles.

- bool isZConstant () const

  Check if the Z coordinate is set to a constant value for all particles.
- virtual void writeParticle (Particle particle)

  Write a particle to the phase space file.

Static Public Member Functions

- static std::vector< CLICommand > getFormatSpecificCLICommands ()

  Get format-specific CLI commands for ROOT configuration.

Static Public Member Functions inherited from ParticleZoo::PhaseSpaceFileWriter

- static std::vector< CLICommand > getCLICommands ()

  Get command line interface commands supported by this writer.

Protected Member Functions

- void writeHeaderData (ByteBuffer &buffer) override

  Write header data (not used for ROOT format).
- void writeParticleManually (Particle &particle) override

  Write particle data to ROOT TTree.

Protected Member Functions inherited from ParticleZoo::PhaseSpaceFileWriter

- virtual bool accountForAdditionalHistories (std::uint64_t additionalHistories)

  Handle accounting for simulation histories that produced no particles.
- virtual bool canHaveConstantPx () const

  Check if this format supports constant X-component of the direction unit vector.
- virtual bool canHaveConstantPy () const

  Check if this format supports constant Y-component of the direction unit vector.
- virtual bool canHaveConstantPz () const

  Check if this format supports constant Z-component of the direction unit vector.
- virtual bool canHaveConstantWeight () const

  Check if this format supports constant statistical weights.
- virtual bool canHaveConstantX () const

  Check if this format supports constant X coordinates.
- virtual bool canHaveConstantY () const

  Check if this format supports constant Y coordinates.
- virtual bool canHaveConstantZ () const

Check if this format supports constant Z coordinates.
- virtual bool canWritePseudoParticlesExplicitly () const

  Check if this format can write pseudo-particles explicitly.
- virtual void fixedValuesHaveChanged ()

  Called when fixed values have been changed.
- virtual size_t getMaximumASCIILineLength () const

  Get the maximum line length for ASCII format files.
- virtual std::size_t getParticleRecordLength () const

  Get the length in bytes of each particle record.
- virtual std::size_t getParticleRecordStartOffset () const

  Get the byte offset where particle records start in the file.
- virtual std::uint64_t getPendingHistories () const

  Get the number of pending histories to account for.
- const UserOptions & getUserOptions () const

  Get the user options that were passed to the constructor.
- void setByteOrder (ByteOrder byteOrder)

  Set the byte order for binary data writing.
- void setConstantPx (float Px)

  Set a constant X-component of the direction unit vector for all particles.
- void setConstantPy (float Py)

  Set a constant Y-component of the direction unit vector for all particles.
- void setConstantPz (float Pz)

  Set a constant Z-component of the direction unit vector for all particles.
- void setConstantWeight (float weight)

  Set a constant statistical weight for all particles.
- void setConstantX (float X)

  Set a constant X coordinate value for all particles.
- void setConstantY (float Y)

  Set a constant Y coordinate value for all particles.
- void setConstantZ (float Z)

  Set a constant Z coordinate value for all particles.
- virtual const std::string writeASCIIParticle (Particle &particle)

  Write a particle in ASCII format as a string.
- virtual void writeBinaryParticle (ByteBuffer &buffer, Particle &particle)

  Write a particle in binary format to a byte buffer.

6.16.1 Detailed Description

ROOT format phase space file writer.

Writes particle data to ROOT TTree structures with configurable branch mappings. Supports multiple format presets (TOPAS, OpenGATE) and custom branch configurations.

6.16.2 Constructor & Destructor Documentation

6.16.2.1 Writer()

```
ParticleZoo::ROOT::Writer::Writer (
            const std::string & fileName,
            const UserOptions & options = UserOptions{} )
```

Construct a ROOT file writer with user options.

Parameters

fileName	Path for the output ROOT file
options	User configuration options including branch mappings

Exceptions

std::runtime_error	If file cannot be created or TTree setup fails

6.16.3 Member Function Documentation

6.16.3.1 getFormatSpecificCLICommands()

```
std::vector< CLICommand > ParticleZoo::ROOT::Writer::getFormatSpecificCLICommands ( )  [static]
```

Get format-specific CLI commands for ROOT configuration.

Returns

Vector of ROOT-specific CLI commands

6.16.3.2 getMaximumSupportedParticles()

```
std::uint64_t ParticleZoo::ROOT::Writer::getMaximumSupportedParticles ( ) const  [inline], [override],
[virtual]
```

Get maximum number of particles that can be stored.

Returns

Maximum supported particles

Implements ParticleZoo::PhaseSpaceFileWriter.

6.16.3.3 writeHeaderData()

```
void ParticleZoo::ROOT::Writer::writeHeaderData (
             ByteBuffer & buffer )  [inline], [override], [protected], [virtual]
```

Write header data (not used for ROOT format).

Parameters

buffer	Unused buffer parameter

Implements ParticleZoo::PhaseSpaceFileWriter.

6.16.3.4 writeParticleManually()

```
void ParticleZoo::ROOT::Writer::writeParticleManually (
             Particle & particle )  [override], [protected], [virtual]
```

Write particle data to ROOT TTree.

Parameters

particle	Particle object to write to current TTree entry

Reimplemented from ParticleZoo::PhaseSpaceFileWriter.

The documentation for this class was generated from the following files:

- include/particlezoo/ROOT/ROOTphsp.h
- src/ROOT/ROOTphsp.cc

6.17 ParticleZoo::SupportedFormat Struct Reference

Structure describing a supported phase space file format.

```
#include <particlezoo/utilities/formats.h>
```

Public Attributes

- const std::string **description**

  Human-readable description of the format.
- const std::string **fileExtension**

  Standard file extension (e.g., ".egsphsp", ".IAEAphsp")
- const bool **fileExtensionCanHaveSuffix** {false}

  True if extension can have numeric suffixes (e.g., ".egsphsp1")
- const std::string **name**

  Format name (e.g., "IAEA", "EGS", "TOPAS")

6.17.1 Detailed Description

Structure describing a supported phase space file format.

Contains metadata about a file format including its name, description, file extension, and whether the extension can have numeric suffixes.

The documentation for this struct was generated from the following file:

- include/particlezoo/utilities/formats.h

6.18 ParticleZoo::TOPASphspFile::Header Class Reference

Header for TOPAS phase space files.

```
#include <particlezoo/TOPAS/TOPASHeader.h>
```

Classes

- struct DataColumn

  Column definition for TOPAS phase space files.
- struct ParticleStats

  Statistics tracking for individual particle types for TOPAS phase space files.

Public Types

- enum class ColumnType {
  POSITION_X , POSITION_Y , POSITION_Z , DIRECTION_COSINE_X ,
  DIRECTION_COSINE_Y , ENERGY , WEIGHT , PARTICLE_TYPE ,
  DIRECTION_COSINE_Z_SIGN , NEW_HISTORY_FLAG , TOPAS_TIME , TIME_OF_FLIGHT ,
  RUN_ID , EVENT_ID , TRACK_ID , PARENT_ID ,
  CHARGE , CREATOR_PROCESS , INITIAL_KINETIC_ENERGY , VERTEX_POSITION_X ,
  VERTEX_POSITION_Y , VERTEX_POSITION_Z , INITIAL_DIRECTION_COSINE_X , INITIAL_DIRECTION_COSINE_Y
  ,
  INITIAL_DIRECTION_COSINE_Z , SEED_PART_1 , SEED_PART_2 , SEED_PART_3 ,
  SEED_PART_4 }

  Column types supported in TOPAS phase space files.
- enum class DataType {
  STRING , BOOLEAN , INT8 , INT32 ,
  FLOAT32 , FLOAT64 }

  Data types supported in TOPAS columns.
- using **ParticleStatsTable** = std::unordered_map< ParticleType, ParticleStats >

  Type alias for particle statistics table.

Public Member Functions

- Header (const std::string &fileName)

  Construct header by reading from existing TOPAS file.
- Header (const std::string &fileName, TOPASFormat formatType)

  Construct header for writing new TOPAS file.
- void addColumnType (ColumnType columnType)

  Add a new column type to the phase space format.

- void countParticleStats (const Particle &particle)

  Update particle statistics with a new particle.
- const std::vector< DataColumn > & getColumnTypes () const

  Get the column definitions for this phase space.
- double getMaxKineticEnergyOfType (ParticleType type) const

  Get the maximum kinetic energy for particles of a specific type.
- double getMinKineticEnergyOfType (ParticleType type) const

  Get the minimum kinetic energy for particles of a specific type.
- std::uint64_t getNumberOfOriginalHistories () const

  Get the number of original simulation histories.
- std::uint64_t getNumberOfParticles () const

  Get the total number of particles in the phase space.
- std::uint64_t getNumberOfParticlesOfType (ParticleType type) const

  Get the number of particles of a specific type.
- std::uint64_t getNumberOfRepresentedHistories () const

  Get the number of histories explicitly represented by particles in the phase space.
- std::size_t getRecordLength () const

  Get the length of each particle record in bytes.
- TOPASFormat getTOPASFormat () const

  Get the TOPAS format type.
- std::string getTOPASFormatName () const

  Get the human-readable format name.
- void setNumberOfOriginalHistories (std::uint64_t originalHistories)

  Set the number of original simulation histories.
- void writeHeader ()

  Write the complete header file.

Static Public Member Functions

- static std::string getTOPASFormatName (TOPASFormat format)

  Get format name from enum value.

6.18.1 Detailed Description

Header for TOPAS phase space files.

This class handles reading, writing, and managing header information for TOPAS format phase space files. It manages file metadata, particle statistics, column definitions, and format-specific configurations. TOPAS files use separate .header and .phsp files.

6.18.2 Member Enumeration Documentation

6.18.2.1 ColumnType

`enum class ParticleZoo::TOPASphspFile::Header::ColumnType [strong]`

Column types supported in TOPAS phase space files.

Defines all possible column types that can appear in TOPAS files, from basic particle properties to extended tracking information.

Enumerator

POSITION_X	X coordinate position.
POSITION_Y	Y coordinate position.
POSITION_Z	Z coordinate position.
DIRECTION_COSINE_X	X direction cosine.
DIRECTION_COSINE_Y	Y direction cosine.
ENERGY	Kinetic energy.
WEIGHT	Particle statistical weight.
PARTICLE_TYPE	PDG particle type code.
DIRECTION_COSINE_Z_SIGN	Sign of Z direction cosine.
NEW_HISTORY_FLAG	First particle from history flag.
TOPAS_TIME	TOPAS simulation time.
TIME_OF_FLIGHT	Particle time of flight.
RUN_ID	Simulation run identifier.
EVENT_ID	Event identifier within run.
TRACK_ID	Track identifier within event.
PARENT_ID	Parent track identifier.
CHARGE	Particle charge.
CREATOR_PROCESS	Physics process that created particle.
INITIAL_KINETIC_ENERGY	Initial kinetic energy at creation.
VERTEX_POSITION_X	X coordinate of creation vertex.
VERTEX_POSITION_Y	Y coordinate of creation vertex.
VERTEX_POSITION_Z	Z coordinate of creation vertex.
INITIAL_DIRECTION_COSINE_X	Initial X direction cosine.
INITIAL_DIRECTION_COSINE_Y	Initial Y direction cosine.
INITIAL_DIRECTION_COSINE_Z	Initial Z direction cosine.
SEED_PART_1	Random seed component 1.
SEED_PART_2	Random seed component 2.
SEED_PART_3	Random seed component 3.
SEED_PART_4	Random seed component 4.

6.18.2.2 DataType

```
enum class ParticleZoo::TOPASphspFile::Header::DataType  [strong]
```

Data types supported in TOPAS columns.

Defines the fundamental data types that can be stored in TOPAS phase space file columns.

Enumerator

STRING	Variable-length string data.
BOOLEAN	Boolean true/false values.
INT8	8-bit signed integer
INT32	32-bit signed integer
FLOAT32	32-bit floating-point
FLOAT64	64-bit floating-point

6.18.3 Constructor & Destructor Documentation

6.18.3.1 Header() [1/2]

```
ParticleZoo::TOPASphspFile::Header::Header (
             const std::string & fileName )
```

Construct header by reading from existing TOPAS file.

Parameters

fileName	Path to TOPAS phase space file (.phsp or .header)

Exceptions

std::runtime_error	if file cannot be read or is invalid

6.18.3.2 Header() [2/2]

```
ParticleZoo::TOPASphspFile::Header::Header (
             const std::string & fileName,
             TOPASFormat formatType )
```

Construct header for writing new TOPAS file.

Parameters

fileName	Path for new TOPAS phase space file
formatType	Format to write (ASCII, BINARY, or LIMITED)

6.18.4 Member Function Documentation

6.18.4.1 addColumnType()

```
void ParticleZoo::TOPASphspFile::Header::addColumnType (
             ColumnType columnType )
```

Add a new column type to the phase space format.

Parameters

columnType	Type of column to add

6.18.4.2 countParticleStats()

```
void ParticleZoo::TOPASphspFile::Header::countParticleStats (
             const Particle & particle )  [inline]
```

Update particle statistics with a new particle.

Parameters

particle	Particle to include in statistics calculations

6.18.4.3 getColumnTypes()

```
const std::vector< Header::DataColumn > & ParticleZoo::TOPASphspFile::Header::getColumnTypes ( )
const  [inline]
```

Get the column definitions for this phase space.

Returns

Vector of DataColumn objects describing the file structure

6.18.4.4 getMaxKineticEnergyOfType()

```
double ParticleZoo::TOPASphspFile::Header::getMaxKineticEnergyOfType (
             ParticleType type ) const  [inline]
```

Get the maximum kinetic energy for particles of a specific type.

Parameters

type	Particle type to query

Returns

Maximum kinetic energy for the particle type

6.18.4.5 getMinKineticEnergyOfType()

```
double ParticleZoo::TOPASphspFile::Header::getMinKineticEnergyOfType (
             ParticleType type ) const  [inline]
```

Get the minimum kinetic energy for particles of a specific type.

Parameters

type	Particle type to query

Returns

Minimum kinetic energy for the particle type

6.18.4.6 getNumberOfOriginalHistories()

```
std::uint64_t ParticleZoo::TOPASphspFile::Header::getNumberOfOriginalHistories ( ) const  [inline]
```

Get the number of original simulation histories.

Returns

Count of primary histories used in the simulation

6.18.4.7 getNumberOfParticles()

```
std::uint64_t ParticleZoo::TOPASphspFile::Header::getNumberOfParticles ( ) const  [inline]
```

Get the total number of particles in the phase space.

Returns

Total particle count across all types

6.18.4.8 getNumberOfParticlesOfType()

```
std::uint64_t ParticleZoo::TOPASphspFile::Header::getNumberOfParticlesOfType (
            ParticleType type ) const  [inline]
```

Get the number of particles of a specific type.

Parameters

type	Particle type to query

Returns

Number of particles of the specified type

6.18.4.9 getNumberOfRepresentedHistories()

```
std::uint64_t ParticleZoo::TOPASphspFile::Header::getNumberOfRepresentedHistories ( ) const  [inline]
```

Get the number of histories explicitly represented by particles in the phase space.

Returns

Count of histories that produced particles that reached the phase space

6.18.4.10 getRecordLength()

```
std::size_t ParticleZoo::TOPASphspFile::Header::getRecordLength ( ) const
```

Get the length of each particle record in bytes.

Returns

Record length based on format and column configuration

6.18.4.11 getTOPASFormat()

```
TOPASFormat ParticleZoo::TOPASphspFile::Header::getTOPASFormat ( ) const  [inline]
```

Get the TOPAS format type.

Returns

TOPASFormat enum value

6.18.4.12 getTOPASFormatName() [1/2]

```
std::string ParticleZoo::TOPASphspFile::Header::getTOPASFormatName ( ) const  [inline]
```

Get the human-readable format name.

Returns

Format name as string (e.g., "TOPAS BINARY")

6.18.4.13 getTOPASFormatName() [2/2]

```
std::string ParticleZoo::TOPASphspFile::Header::getTOPASFormatName (
            TOPASFormat format )  [inline], [static]
```

Get format name from enum value.

Parameters

format	TOPAS format type

Returns

Human-readable format name

6.18.4.14 setNumberOfOriginalHistories()

```
void ParticleZoo::TOPASphspFile::Header::setNumberOfOriginalHistories (
            std::uint64_t originalHistories )  [inline]
```

Set the number of original simulation histories.

Parameters

originalHistories	Count of primary histories

6.18.4.15 writeHeader()

```
void ParticleZoo::TOPASphspFile::Header::writeHeader ( )
```

Write the complete header file.

Writes the header information to the .header file with format-specific structure and particle statistics.

Exceptions

std::runtime_error	if file cannot be written

The documentation for this class was generated from the following files:

- include/particlezoo/TOPAS/TOPASHeader.h
- src/topas/TOPASHeader.cc

6.19 ParticleZoo::TOPASphspFile::Header::DataColumn Struct Reference

Column definition for TOPAS phase space files.

```
#include <particlezoo/TOPAS/TOPASHeader.h>
```

Public Member Functions

- DataColumn (ColumnType columnType)

  Construct column from type (uses default name)
- DataColumn (ColumnType columnType, DataType valueType)

  Construct column with specific data type.
- DataColumn (ColumnType columnType, DataType valueType, std::string_view name)

  Construct column with all parameters specified.
- DataColumn (std::string_view name)

  Construct column from name string.
- std::size_t sizeOf () const

  Get the storage size of this column's data type.

Static Public Member Functions

- static constexpr std::string_view getColumnName (ColumnType columnType)

  Get the string represented name for a column type.
- static constexpr ColumnType getColumnType (std::string_view name)

  Parse column type from name.
- static constexpr DataType getDataType (ColumnType columnType)

  Get the default data type for a column type.

Public Attributes

- ColumnType **columnType_**

  Semantic type of the column.
- std::string **name_**

  Human-readable column name.
- DataType **valueType_**

  Data storage type.

6.19.1 Detailed Description

Column definition for TOPAS phase space files.

Describes a single column in the phase space file including its type, data format, and display name.

6.19.2 Constructor & Destructor Documentation

6.19.2.1 DataColumn() [1/4]

```
ParticleZoo::TOPASphspFile::Header::DataColumn::DataColumn (
            std::string_view name )  [inline]
```

Construct column from name string.

Parameters

name	Column name (determines type automatically)

6.19.2.2 DataColumn() [2/4]

```
ParticleZoo::TOPASphspFile::Header::DataColumn::DataColumn (
            ColumnType columnType )  [inline]
```

Construct column from type (uses default name)

Parameters

columnType	Column type

6.19.2.3 DataColumn() [3/4]

```
ParticleZoo::TOPASphspFile::Header::DataColumn::DataColumn (
            ColumnType columnType,
            DataType valueType )  [inline]
```

Construct column with specific data type.

Parameters

columnType	Column type
valueType	Data storage type (overrides default)

6.19.2.4 DataColumn() `[4/4]`

```
ParticleZoo::TOPASphspFile::Header::DataColumn::DataColumn (
            ColumnType columnType,
            DataType valueType,
            std::string_view name )  [inline]
```

Construct column with all parameters specified.

Parameters

columnType	Column type
valueType	Data storage type
name	Custom column name

6.19.3 Member Function Documentation

6.19.3.1 getColumnName()

```
static constexpr std::string_view ParticleZoo::TOPASphspFile::Header::DataColumn::getColumnName (
            ColumnType columnType )  [inline], [static], [constexpr]
```

Get the string represented name for a column type.

Parameters

columnType	Column type to query

Returns

Human-readable column name with units

Exceptions

std::runtime_error	if column type is unknown

6.19.3.2 getColumnType()

```
static constexpr ColumnType ParticleZoo::TOPASphspFile::Header::DataColumn::getColumnType (
            std::string_view name ) [inline], [static], [constexpr]
```

Parse column type from name.

Parameters

name	Column name to parse

Returns

Corresponding ColumnType

Exceptions

std::runtime_error	if name is not recognized

6.19.3.3 getDataType()

```
static constexpr DataType ParticleZoo::TOPASphspFile::Header::DataColumn::getDataType (
            ColumnType columnType ) [inline], [static], [constexpr]
```

Get the default data type for a column type.

Parameters

columnType	Column type to query

Returns

Default DataType for storage

Exceptions

std::runtime_error	if column type is unknown

6.19.3.4 sizeOf()

```
std::size_t ParticleZoo::TOPASphspFile::Header::DataColumn::sizeOf ( ) const  [inline]
```

Get the storage size of this column's data type.

Returns

Size in bytes (0 for variable-length strings)

The documentation for this struct was generated from the following file:

- include/particlezoo/TOPAS/TOPASHeader.h

6.20 ParticleZoo::TOPASphspFile::Header::ParticleStats Struct Reference

Statistics tracking for individual particle types for TOPAS phase space files.

```
#include <particlezoo/TOPAS/TOPASHeader.h>
```

Public Attributes

- std::uint64_t **count_** {}

    Number of particles of this type.
- double **maxKineticEnergy_** = 0

    Maximum kinetic energy encountered.
- double **minKineticEnergy_** = std::numeric_limits<double>::max()

    Minimum kinetic energy encountered.

6.20.1 Detailed Description

Statistics tracking for individual particle types for TOPAS phase space files.

The documentation for this struct was generated from the following file:

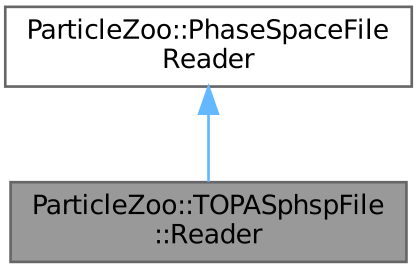
- include/particlezoo/TOPAS/TOPASHeader.h

6.21 ParticleZoo::TOPASphspFile::Reader Class Reference

Reader for TOPAS phase space files.

```
#include <particlezoo/TOPAS/TOPASphspFile.h>
```

Inheritance diagram for ParticleZoo::TOPASphspFile::Reader:


ParticleZoo::PhaseSpaceFile Reader
ParticleZoo::TOPASphspFile ::Reader

Public Member Functions

- Reader (const std::string &filename, const UserOptions &options=UserOptions{})

  Construct reader for TOPAS phase space file.
- const Header & getHeader () const

  Get access to the TOPAS header information.
- std::uint64_t getNumberOfOriginalHistories () const override

  Get the number of original simulation histories.
- std::uint64_t getNumberOfParticles () const override

  Get the total number of particles in the phase space.
- TOPASFormat getTOPASFormat () const

  Get the TOPAS format type of this file.
- void setDetailedReading (bool enable)

  Enable or disable detailed reading of extended particle properties.

Public Member Functions inherited from ParticleZoo::PhaseSpaceFileReader

- PhaseSpaceFileReader (const std::string &phspFormat, const std::string &fileName, const UserOptions &user↩ Options, FormatType formatType=FormatType::BINARY, const FixedValues fixedValues=FixedValues(), unsigned int bufferSize=DEFAULT_BUFFER_SIZE)

  Construct a new Phase Space File Reader object.
- virtual ∼PhaseSpaceFileReader ()

  Destroy the Phase Space File Reader object.
- void close ()

  Close the phase space file and clean up resources.
- float getConstantPx () const

  Get the constant X-component of the direction unit vector (if constant).
- float getConstantPy () const

  Get the constant Y-component of the direction unit vector (if constant).
- float getConstantPz () const

  Get the constant Z-component of the direction unit vector (if constant).
- float getConstantWeight () const

  Get the constant statistical weight value (if constant).
- float getConstantX () const

  Get the constant X coordinate value (if constant).
- float getConstantY () const

  Get the constant Y coordinate value (if constant).
- float getConstantZ () const

  Get the constant Z coordinate value (if constant).
- const std::string getFileName () const

  Get the filename of the phase space file being read.
- std::uint64_t getFileSize () const

  Get the size of the phase space file in bytes.
- const FixedValues getFixedValues () const

  Get the fixed values configuration.
- virtual std::uint64_t getHistoriesRead ()

  Get the number of Monte Carlo histories that have been read so far. If the end of the file has been reached, this will return the total number of original histories unless more histories than expected have already been read - in which case it returns the actual count.
- Particle getNextParticle ()

  Get the next particle from the phase space file.
- virtual std::uint64_t getParticlesRead ()

  Get the number of particles that have been read so far.
- const std::string getPHSPFormat () const

  Get the phase space file format identifier.
- virtual bool hasMoreParticles ()

  Check if there are more particles to read in the file.

- bool isPxConstant () const

  Check if the X-component of momentum is constant for all particles.
- bool isPyConstant () const

  Check if the Y-component of momentum is constant for all particles.
- bool isPzConstant () const

  Check if the Z-component of momentum is constant for all particles.
- bool isWeightConstant () const

  Check if the statistical weight is constant for all particles.
- bool isXConstant () const

  Check if the X coordinate is constant for all particles.
- bool isYConstant () const

  Check if the Y coordinate is constant for all particles.
- bool isZConstant () const

  Check if the Z coordinate is constant for all particles.
- void moveToParticle (std::uint64_t particleIndex)

  Move the file position to a specific particle index.
- void setCommentMarkers (const std::vector< std::string > &commentMarkers)

  Set comment markers for ASCII format files.

Static Public Member Functions

- static std::vector< CLICommand > getFormatSpecificCLICommands ()

  Get format-specific command-line options.

Static Public Member Functions inherited from ParticleZoo::PhaseSpaceFileReader

- static std::vector< CLICommand > getCLICommands ()

  Get command line interface commands supported by this reader.

Protected Member Functions

- std::size_t getMaximumASCIILineLength () const override

  Get the maximum length of ASCII particle lines.
- std::size_t getParticleRecordLength () const override

  Get the length of each particle record in bytes.
- Particle readASCIIParticle (const std::string &line) override

  Parse a single ASCII line into a Particle object.
- Particle readBinaryParticle (ByteBuffer &buffer) override

  Read and decode a single particle from binary data.

Protected Member Functions inherited from ParticleZoo::PhaseSpaceFileReader

- double calcThirdUnitComponent (double &u, double &v) const

  Calculate the third component of a unit vector from two components (double precision).
- float calcThirdUnitComponent (float &u, float &v) const

  Calculate the third component of a unit vector from two components (float precision).
- const ByteBuffer getHeaderData ()

  Get the file header data as a byte buffer.
- const ByteBuffer getHeaderData (std::size_t headerSize)

  Get a specific amount of header data as a byte buffer.
- Particle getNextParticle (bool countParticleInStatistics)

  Get the next particle with optional statistics counting control.
- std::size_t getNumberOfEntriesInFile () const

  Get the number of particle records that fit in the file.
- virtual std::size_t getParticleRecordStartOffset () const

  Get the byte offset where particle records start in the file.
- virtual std::uint64_t getParticlesRead (bool includeSkippedParticles)

  Get the number of particles read with optional inclusion of skipped particles (including pseudo-particles).
- const UserOptions & getUserOptions () const

  Get the user options that were passed to the constructor.
- virtual Particle readParticleManually ()

  Read a particle manually (for formats requiring third-party I/O).
- void setByteOrder (ByteOrder byteOrder)

  Set the byte order for binary data interpretation.
- void setConstantPx (float Px)

  Set a constant X-component of the direction unit vector for all particles.
- void setConstantPy (float Py)

  Set a constant Y-component of the direction unit vector for all particles.
- void setConstantPz (float Pz)

  Set a constant Z-component of the direction unit vector for all particles.
- void setConstantWeight (float weight)

  Set a constant statistical weight for all particles.
- void setConstantX (float X)

  Set a constant X coordinate value for all particles.
- void setConstantY (float Y)

  Set a constant Y coordinate value for all particles.
- void setConstantZ (float Z)

  Set a constant Z coordinate value for all particles.

6.21.1 Detailed Description

Reader for TOPAS phase space files.

Provides functionality to read phase space data from TOPAS format files, supporting ASCII, BINARY, and LIMITED formats. Handles TOPAS-specific features including pseudo-particles for empty history tracking and extensive metadata columns.

6.21.2 Constructor & Destructor Documentation

6.21.2.1 Reader()

```
ParticleZoo::TOPASphspFile::Reader::Reader (
             const std::string & filename,
             const UserOptions & options = UserOptions{} )
```

Construct reader for TOPAS phase space file.

Automatically detects the format (ASCII, BINARY, or LIMITED) by reading the header file and configures the reader accordingly.

Parameters

filename	Path to the TOPAS phase space file (.phsp or .header)
options	User-specified options for reading behavior

Exceptions

std::runtime_error	if file cannot be opened or format is invalid

6.21.3 Member Function Documentation

6.21.3.1 getFormatSpecificCLICommands()

```
std::vector< CLICommand > ParticleZoo::TOPASphspFile::Reader::getFormatSpecificCLICommands ( )
[static]
```

Get format-specific command-line options.

Returns

Vector of CLI commands supported by TOPAS reader (currently empty)

6.21.3.2 getHeader()

```
const Header & ParticleZoo::TOPASphspFile::Reader::getHeader ( ) const  [inline]
```

Get access to the TOPAS header information.

Returns

Reference to the header containing file metadata and column definitions

6.21.3.3 getMaximumASCIILineLength()

```
std::size_t ParticleZoo::TOPASphspFile::Reader::getMaximumASCIILineLength ( ) const  [inline],
[override], [protected], [virtual]
```

Get the maximum length of ASCII particle lines.

Returns

Maximum line length (1024 characters for TOPAS format)

Reimplemented from ParticleZoo::PhaseSpaceFileReader.

6.21.3.4 getNumberOfOriginalHistories()

```
std::uint64_t ParticleZoo::TOPASphspFile::Reader::getNumberOfOriginalHistories ( ) const  [inline],
[override], [virtual]
```

Get the number of original simulation histories.

Returns

Count of primary histories used in the simulation

Implements ParticleZoo::PhaseSpaceFileReader.

6.21.3.5 getNumberOfParticles()

```
std::uint64_t ParticleZoo::TOPASphspFile::Reader::getNumberOfParticles ( ) const  [inline], [override],
[virtual]
```

Get the total number of particles in the phase space.

Returns

Total particle count as recorded in the header

Implements ParticleZoo::PhaseSpaceFileReader.

6.21.3.6 getParticleRecordLength()

```
std::size_t ParticleZoo::TOPASphspFile::Reader::getParticleRecordLength ( ) const  [inline], [override],
[protected], [virtual]
```

Get the length of each particle record in bytes.

Returns

Record length as defined by the TOPAS header

Reimplemented from ParticleZoo::PhaseSpaceFileReader.

6.21.3.7 getTOPASFormat()

```
TOPASFormat ParticleZoo::TOPASphspFile::Reader::getTOPASFormat ( ) const  [inline]
```

Get the TOPAS format type of this file.

Returns

TOPASFormat enum indicating ASCII, BINARY, or LIMITED

6.21.3.8 readASCIIParticle()

```
Particle ParticleZoo::TOPASphspFile::Reader::readASCIIParticle (
             const std::string & line )  [override], [protected], [virtual]
```

Parse a single ASCII line into a Particle object.

Parses TOPAS ASCII format with configurable columns. Supports both core particle data and extended properties based on header column definitions.

Parameters

line	ASCII line containing particle data

Returns

Parsed Particle object with properties set according to column types

Exceptions

std::runtime_error	if line cannot be parsed or contains invalid data

Reimplemented from ParticleZoo::PhaseSpaceFileReader.

6.21.3.9 readBinaryParticle()

```
Particle ParticleZoo::TOPASphspFile::Reader::readBinaryParticle (
            ByteBuffer & buffer )  [inline], [override], [protected], [virtual]
```

Read and decode a single particle from binary data.

Handles format-specific binary reading including pseudo-particle processing for empty history tracking in BINARY format.

Parameters

buffer	Binary buffer containing particle data

Returns

Decoded Particle object with all properties

Exceptions

std::runtime_error	if format is unsupported or data is corrupted

Reimplemented from ParticleZoo::PhaseSpaceFileReader.

6.21.3.10 setDetailedReading()

```
void ParticleZoo::TOPASphspFile::Reader::setDetailedReading (
             bool enable )  [inline]
```

Enable or disable detailed reading of extended particle properties.

When enabled, reads all additional columns beyond the standard 10 columns. When disabled, only reads the core particle data for improved performance.

Parameters

enable	true to read all columns, false for core data only

The documentation for this class was generated from the following files:

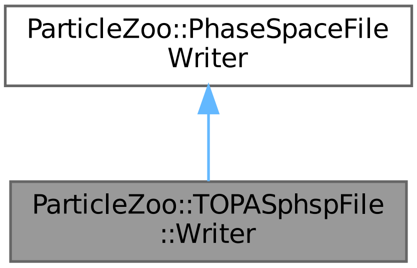
- include/particlezoo/TOPAS/TOPASphspFile.h
- src/topas/TOPASphspFile.cc

6.22 ParticleZoo::TOPASphspFile::Writer Class Reference

Writer for TOPAS phase space files.

```
#include <particlezoo/TOPAS/TOPASphspFile.h>
```

Inheritance diagram for ParticleZoo::TOPASphspFile::Writer:


ParticleZoo::PhaseSpaceFile Writer
ParticleZoo::TOPASphspFile ::Writer

Public Member Functions

- Writer (const std::string &filename, const UserOptions &options=UserOptions{})

  Construct writer for TOPAS phase space file.
- Header & getHeader ()

  Get access to the TOPAS header for configuration.
- std::uint64_t getMaximumSupportedParticles () const override

  Get the maximum number of particles this format can store.
- TOPASFormat getTOPASFormat () const

  Get the TOPAS format type being written.

Public Member Functions inherited from ParticleZoo::PhaseSpaceFileWriter

- PhaseSpaceFileWriter (const std::string &phspFormat, const std::string &fileName, const UserOptions &user←Options, FormatType formatType=FormatType::BINARY, const FixedValues fixedValues=FixedValues(), unsigned int bufferSize=DEFAULT_BUFFER_SIZE)

  Construct a new Phase Space File Writer object.
- virtual ∼PhaseSpaceFileWriter ()

Destroy the Phase Space File Writer object.

- void addAdditionalHistories (std::uint64_t additionalHistories)

  Add additional Monte Carlo histories to the count.
- void close ()

  Close the phase space file and finalize writing.
- ByteOrder getByteOrder () const

  Get the byte order used for binary data writing.
- float getConstantPx () const

  Get the constant X-component of the direction unit vector (if constant).
- float getConstantPy () const

  Get the constant Y-component of the direction unit vector (if constant).
- float getConstantPz () const

  Get the constant Z-component of the direction unit vector (if constant).
- float getConstantWeight () const

  Get the constant statistical weight value (if constant).
- float getConstantX () const

  Get the constant X coordinate value (if constant).
- float getConstantY () const

  Get the constant Y coordinate value (if constant).
- float getConstantZ () const

  Get the constant Z coordinate value (if constant).
- const std::string getFileName () const

  Get the filename where the phase space file is being written.
- const FixedValues getFixedValues () const

  Get the fixed values configuration.
- virtual std::uint64_t getHistoriesWritten () const

  Get the number of Monte Carlo histories that have been written.
- std::uint64_t getParticlesWritten () const

  Get the number of particles that have been written to the file.
- const std::string getPHSPFormat () const

  Get the phase space file format identifier.
- bool isPxConstant () const

  Check if the X-component of the direction unit vector is set to a constant value for all particles.
- bool isPyConstant () const

  Check if the Y-component of the direction unit vector is set to a constant value for all particles.
- bool isPzConstant () const

  Check if the Z-component of the direction unit vector is set to a constant value for all particles.
- bool isWeightConstant () const

  Check if the statistical weight is set to a constant value for all particles.
- bool isXConstant () const

  Check if the X coordinate is set to a constant value for all particles.

- bool isYConstant () const

  Check if the Y coordinate is set to a constant value for all particles.
- bool isZConstant () const

  Check if the Z coordinate is set to a constant value for all particles.
- virtual void writeParticle (Particle particle)

  Write a particle to the phase space file.

Static Public Member Functions

- static std::vector< CLICommand > getFormatSpecificCLICommands ()

  Get format-specific command-line options.

Static Public Member Functions inherited from ParticleZoo::PhaseSpaceFileWriter

- static std::vector< CLICommand > getCLICommands ()

  Get command line interface commands supported by this writer.

Protected Member Functions

- bool accountForAdditionalHistories (std::uint64_t additionalHistories) override

  Override base class method to handle additional histories.
- bool canWritePseudoParticlesExplicitly () const override

  Check if pseudo-particles can be written explicitly.
- std::size_t getMaximumASCIILineLength () const override

  Get the maximum length of ASCII particle lines.
- std::size_t getParticleRecordLength () const override

  Get the length of each particle record in bytes.
- std::uint64_t getPendingHistories () const override

  Get the number of pending histories to account for.
- const std::string writeASCIIParticle (Particle &particle) override

  Convert a particle to ASCII representation.
- void writeBinaryParticle (ByteBuffer &buffer, Particle &particle) override

  Encode and write a single particle to binary data.
- void writeHeaderData (ByteBuffer &buffer) override

  Write header data to the output buffer.

Protected Member Functions inherited from ParticleZoo::PhaseSpaceFileWriter

- virtual bool canHaveConstantPx () const

  Check if this format supports constant X-component of the direction unit vector.
- virtual bool canHaveConstantPy () const

  Check if this format supports constant Y-component of the direction unit vector.
- virtual bool canHaveConstantPz () const

  Check if this format supports constant Z-component of the direction unit vector.
- virtual bool canHaveConstantWeight () const

  Check if this format supports constant statistical weights.
- virtual bool canHaveConstantX () const

  Check if this format supports constant X coordinates.
- virtual bool canHaveConstantY () const

  Check if this format supports constant Y coordinates.
- virtual bool canHaveConstantZ () const

  Check if this format supports constant Z coordinates.
- virtual void fixedValuesHaveChanged ()

  Called when fixed values have been changed.
- virtual std::size_t getParticleRecordStartOffset () const

  Get the byte offset where particle records start in the file.
- const UserOptions & getUserOptions () const

  Get the user options that were passed to the constructor.
- void setByteOrder (ByteOrder byteOrder)

  Set the byte order for binary data writing.
- void setConstantPx (float Px)

  Set a constant X-component of the direction unit vector for all particles.
- void setConstantPy (float Py)

  Set a constant Y-component of the direction unit vector for all particles.
- void setConstantPz (float Pz)

  Set a constant Z-component of the direction unit vector for all particles.
- void setConstantWeight (float weight)

  Set a constant statistical weight for all particles.
- void setConstantX (float X)

  Set a constant X coordinate value for all particles.
- void setConstantY (float Y)

  Set a constant Y coordinate value for all particles.
- void setConstantZ (float Z)

  Set a constant Z coordinate value for all particles.
- virtual void writeParticleManually (Particle &particle)

  Write a particle manually (for formats requiring third-party I/O).

6.22.1 Detailed Description

Writer for TOPAS phase space files.

Provides functionality to write phase space data in TOPAS format files, supporting ASCII, BINARY, and LIMITED formats. Handles TOPAS-specific features including pseudo-particle generation for efficient empty history tracking and configurable column layouts.

6.22.2 Constructor & Destructor Documentation

6.22.2.1 Writer()

```
ParticleZoo::TOPASphspFile::Writer::Writer (
             const std::string & filename,
             const UserOptions & options = UserOptions{} )
```

Construct writer for TOPAS phase space file.

Creates a new TOPAS format writer with format determined by command-line options. Defaults to BINARY format if no format is specified.

Parameters

filename	Path for the output TOPAS phase space file (.phsp)
options	User-specified options including format selection

Exceptions

std::runtime_error	if file cannot be created or format is invalid

6.22.3 Member Function Documentation

6.22.3.1 accountForAdditionalHistories()

```
bool ParticleZoo::TOPASphspFile::Writer::accountForAdditionalHistories (
             std::uint64_t additionalHistories )  [inline], [override], [protected], [virtual]
```

Override base class method to handle additional histories.

For BINARY format, creates a pseudo-particle to represent multiple empty histories in a single record. For other formats, delegates to base class.

Parameters

additionalHistories	Number of additional empty histories to account for

Returns

false for BINARY (handled internally), true for others (use base class)

Reimplemented from ParticleZoo::PhaseSpaceFileWriter.

6.22.3.2 canWritePseudoParticlesExplicitly()

```
bool ParticleZoo::TOPASphspFile::Writer::canWritePseudoParticlesExplicitly ( ) const  [inline],
[override], [protected], [virtual]
```

Check if pseudo-particles can be written explicitly.

Returns

true for BINARY format, false for ASCII and LIMITED

Reimplemented from ParticleZoo::PhaseSpaceFileWriter.

6.22.3.3 getFormatSpecificCLICommands()

```
std::vector< CLICommand > ParticleZoo::TOPASphspFile::Writer::getFormatSpecificCLICommands ( )
[static]
```

Get format-specific command-line options.

Returns

Vector of CLI commands supported by TOPAS writer

6.22.3.4 getHeader()

```
Header & ParticleZoo::TOPASphspFile::Writer::getHeader ( )  [inline]
```

Get access to the TOPAS header for configuration.

Returns

Reference to the header for modification and column management

6.22.3.5 getMaximumASCIILineLength()

`std::size_t ParticleZoo::TOPASphspFile::Writer::getMaximumASCIILineLength ( ) const [inline], [override], [protected], [virtual]`

Get the maximum length of ASCII particle lines.

Returns

Maximum line length (1024 characters for TOPAS format)

Reimplemented from ParticleZoo::PhaseSpaceFileWriter.

6.22.3.6 getMaximumSupportedParticles()

`std::uint64_t ParticleZoo::TOPASphspFile::Writer::getMaximumSupportedParticles ( ) const [inline], [override], [virtual]`

Get the maximum number of particles this format can store.

Returns

Maximum particle count

Implements ParticleZoo::PhaseSpaceFileWriter.

6.22.3.7 getParticleRecordLength()

`std::size_t ParticleZoo::TOPASphspFile::Writer::getParticleRecordLength ( ) const [inline], [override], [protected], [virtual]`

Get the length of each particle record in bytes.

Returns

Record length as defined by the header and format type

Reimplemented from ParticleZoo::PhaseSpaceFileWriter.

6.22.3.8 getPendingHistories()

```
std::uint64_t ParticleZoo::TOPASphspFile::Writer::getPendingHistories ( ) const  [inline], [override],
[protected], [virtual]
```

Get the number of pending histories to account for.

In this override these are the empty histories tracked for pseudo-particle writing.

Returns

Count of empty histories pending writing

Reimplemented from ParticleZoo::PhaseSpaceFileWriter.

6.22.3.9 getTOPASFormat()

```
TOPASFormat ParticleZoo::TOPASphspFile::Writer::getTOPASFormat ( ) const  [inline]
```

Get the TOPAS format type being written.

Returns

TOPASFormat enum indicating ASCII, BINARY, or LIMITED

6.22.3.10 writeASCIIParticle()

```
const std::string ParticleZoo::TOPASphspFile::Writer::writeASCIIParticle (
            Particle & particle )  [override], [protected], [virtual]
```

Convert a particle to ASCII representation.

Formats a particle according to TOPAS ASCII specification with configurable columns.

Parameters

particle	Particle object to convert to ASCII

Returns

ASCII string representation

Exceptions

std::runtime_error	if particle type is unsupported

Reimplemented from ParticleZoo::PhaseSpaceFileWriter.

6.22.3.11 writeBinaryParticle()

```
void ParticleZoo::TOPASphspFile::Writer::writeBinaryParticle (
            ByteBuffer & buffer,
            Particle & particle )  [inline], [override], [protected], [virtual]
```

Encode and write a single particle to binary data.

Handles format-specific binary encoding including pseudo-particle generation for empty history accounting in BINARY format.

Parameters

buffer	Binary buffer to write particle data to
particle	Particle object to encode and store

Exceptions

std::runtime_error	if format is unsupported

Reimplemented from ParticleZoo::PhaseSpaceFileWriter.

6.22.3.12 writeHeaderData()

```
void ParticleZoo::TOPASphspFile::Writer::writeHeaderData (
            ByteBuffer & buffer )  [override], [protected], [virtual]
```

Write header data to the output buffer.

Updates header statistics and writes the complete header file. The header is written to a separate .header file.

Parameters

buffer	Binary buffer (unused for TOPAS as header is separate)

Implements ParticleZoo::PhaseSpaceFileWriter.

The documentation for this class was generated from the following files:

- include/particlezoo/TOPAS/TOPASphspFile.h
- src/topas/TOPASphspFile.cc

6.23 ParticleZoo::Version Struct Reference

Version information and metadata for the ParticleZoo library.

```
#include <particlezoo/utilities/version.h>
```

Static Public Member Functions

- static std::string GetVersionString ()

    Generate a complete version string for display.

Static Public Attributes

- static constexpr const char ∗ CAVEAT = ""

    Development status indicator.
- static constexpr const int MAJOR_VERSION = 1

    Major version number.
- static constexpr const int MINOR_VERSION = 0

    Minor version number.
- static constexpr const int PATCH_VERSION = 0

    Patch version number.
- static constexpr const char ∗ PROJECT_NAME = "ParticleZoo"

    Official project name.

6.23.1 Detailed Description

Version information and metadata for the ParticleZoo library.

This structure provides compile-time constants for version identification and runtime functions for generating version strings. It serves as the authoritative source for all version-related information used throughout the library, documentation generation, and user applications.

The version follows semantic versioning (SemVer) principles with MAJOR.MINOR.PATCH numbering plus development status indicators.

6.23.2 Member Function Documentation

6.23.2.1 GetVersionString()

```
static std::string ParticleZoo::Version::GetVersionString ( )  [inline], [static]
```

Generate a complete version string for display.

Creates a human-readable version string combining all version components in the standard format: "ProjectName vMAJOR.MINOR.PATCH STATUS"

Returns

Complete version string including project name, version numbers, and status

6.23.3 Member Data Documentation

6.23.3.1 CAVEAT

```
constexpr const char* ParticleZoo::Version::CAVEAT = ""  [static], [constexpr]
```

Development status indicator.

Indicates the current development status of this version. Only used to indicate pre-release or special build states.

6.23.3.2 MAJOR_VERSION

```
constexpr const int ParticleZoo::Version::MAJOR_VERSION = 1  [static], [constexpr]
```

Major version number.

Incremented for backwards-incompatible API changes. Applications built against different major versions may require code changes.

6.23.3.3 MINOR_VERSION

```
constexpr const int ParticleZoo::Version::MINOR_VERSION = 0  [static], [constexpr]
```

Minor version number.

Incremented for backwards-compatible feature additions. Applications built against the same major version should work with higher minor versions without modification.

6.23.3.4 PATCH_VERSION

```
constexpr const int ParticleZoo::Version::PATCH_VERSION = 0  [static], [constexpr]
```

Patch version number.

Incremented for backwards-compatible bug fixes and minor improvements that do not add new features. Applications should always be able to upgrade to higher patch versions safely.

6.23.3.5 PROJECT_NAME

```
constexpr const char* ParticleZoo::Version::PROJECT_NAME = "ParticleZoo"  [static], [constexpr]
```

Official project name.

The canonical name of the ParticleZoo library used in all user-facing output, documentation, and version strings.

The documentation for this struct was generated from the following file:

- include/particlezoo/utilities/version.h

Index

∼PhaseSpaceFileReader
    ParticleZoo::PhaseSpaceFileReader, 162
∼PhaseSpaceFileWriter
    ParticleZoo::PhaseSpaceFileWriter, 185

accountForAdditionalHistories
    ParticleZoo::PhaseSpaceFileWriter, 185
    ParticleZoo::TOPASphspFile::Writer, 247
addAdditionalHistories
    ParticleZoo::PhaseSpaceFileWriter, 186
addColumnType
    ParticleZoo::TOPASphspFile::Header, 223
addExtraFloat
    ParticleZoo::IAEAphspFile::IAEAHeader, 83
addExtraLong
    ParticleZoo::IAEAphspFile::IAEAHeader, 83
ADDITIONAL_NOTES
    ParticleZoo::IAEAphspFile::IAEAHeader, 81
appendData
    ParticleZoo::ByteBuffer, 38, 39
ASCII
    ParticleZoo, 29
AUTHORS
    ParticleZoo::IAEAphspFile::IAEAHeader, 81

BEAM_NAME
    ParticleZoo::IAEAphspFile::IAEAHeader, 81
BigEndian
    ParticleZoo, 28
BINARY
    ParticleZoo, 29
BOOLEAN
    ParticleZoo::TOPASphspFile::Header, 222
BoolPropertyType
    ParticleZoo, 28
BYTE_ORDER
    ParticleZoo::IAEAphspFile::IAEAHeader, 81
ByteBuffer
    ParticleZoo::ByteBuffer, 37, 38
ByteOrder
    ParticleZoo, 28

calcThirdUnitComponent
    ParticleZoo::PhaseSpaceFileReader, 162
canHaveConstantPx
    ParticleZoo::IAEAphspFile::Writer, 122
    ParticleZoo::PhaseSpaceFileWriter, 186
canHaveConstantPy
    ParticleZoo::IAEAphspFile::Writer, 122
    ParticleZoo::PhaseSpaceFileWriter, 186
canHaveConstantPz
    ParticleZoo::IAEAphspFile::Writer, 122
    ParticleZoo::PhaseSpaceFileWriter, 187
canHaveConstantWeight
    ParticleZoo::IAEAphspFile::Writer, 122
    ParticleZoo::PhaseSpaceFileWriter, 187
canHaveConstantX
    ParticleZoo::IAEAphspFile::Writer, 123
    ParticleZoo::PhaseSpaceFileWriter, 187
canHaveConstantY
    ParticleZoo::IAEAphspFile::Writer, 123
    ParticleZoo::PhaseSpaceFileWriter, 188
canHaveConstantZ
    ParticleZoo::IAEAphspFile::Writer, 123
    ParticleZoo::PhaseSpaceFileWriter, 188
canWritePseudoParticlesExplicitly
    ParticleZoo::PhaseSpaceFileWriter, 188
    ParticleZoo::TOPASphspFile::Writer, 248
capacity
    ParticleZoo::ByteBuffer, 39
CAVEAT
    ParticleZoo::Version, 254
CHARGE
    ParticleZoo::TOPASphspFile::Header, 221
CHECKSUM
    ParticleZoo::IAEAphspFile::IAEAHeader, 81
checksumIsValid
    ParticleZoo::IAEAphspFile::IAEAHeader, 83

close
    ParticleZoo::PhaseSpaceFileReader, 163
    ParticleZoo::PhaseSpaceFileWriter, 189
ColumnType
    ParticleZoo::TOPASphspFile::Header, 221
compact
    ParticleZoo::ByteBuffer, 39
COORDINATE_SYSTEM_DESCRIPTION
    ParticleZoo::IAEAphspFile::IAEAHeader, 81
countParticleStats
    ParticleZoo::IAEAphspFile::IAEAHeader, 83
    ParticleZoo::TOPASphspFile::Header, 223
CreateReader
    ParticleZoo::FormatRegistry, 68
CreateWriter
    ParticleZoo::FormatRegistry, 69, 70
CREATOR_PROCESS
    ParticleZoo::TOPASphspFile::Header, 221
CUSTOM
    ParticleZoo, 28–30
CUSTOM_SECTION
    ParticleZoo::IAEAphspFile::IAEAHeader, 81

data
    ParticleZoo::ByteBuffer, 40
DataColumn
    ParticleZoo::TOPASphspFile::Header::DataColumn, 229, 230
DataType
    ParticleZoo::TOPASphspFile::Header, 222
DeterminePathToHeaderFile
    ParticleZoo::IAEAphspFile::IAEAHeader, 84
DIRECTION_COSINE_X
    ParticleZoo::TOPASphspFile::Header, 221
DIRECTION_COSINE_Y
    ParticleZoo::TOPASphspFile::Header, 221
DIRECTION_COSINE_Z_SIGN
    ParticleZoo::TOPASphspFile::Header, 221

EGS_LATCH
    ParticleZoo, 29
    ParticleZoo::IAEAphspFile::IAEAHeader, 80
ELECTRONS
    ParticleZoo::IAEAphspFile::IAEAHeader, 81
ENERGY
    ParticleZoo::TOPASphspFile::Header, 221
EVENT_ID
    ParticleZoo::TOPASphspFile::Header, 221
expand
    ParticleZoo::ByteBuffer, 40
ExtensionForFormat
    ParticleZoo::FormatRegistry, 70
EXTRA_FLOAT_TYPE
    ParticleZoo::IAEAphspFile::IAEAHeader, 79
EXTRA_LONG_TYPE
    ParticleZoo::IAEAphspFile::IAEAHeader, 79

FIELD_SIZE
    ParticleZoo::IAEAphspFile::IAEAHeader, 81
FILE_EXTENSION_CAN_HAVE_SUFFIX
    ParticleZoo::FormatRegistry, 73
FILE_TYPE
    ParticleZoo::IAEAphspFile::IAEAHeader, 81
FileType
    ParticleZoo::IAEAphspFile::IAEAHeader, 80
fixedValuesHaveChanged
    ParticleZoo::IAEAphspFile::Writer, 124
    ParticleZoo::PhaseSpaceFileWriter, 189
FLOAT32
    ParticleZoo::TOPASphspFile::Header, 222
FLOAT64
    ParticleZoo::TOPASphspFile::Header, 222
FloatPropertyType
    ParticleZoo, 28
FormatsForExtension
    ParticleZoo::FormatRegistry, 71
FormatType
    ParticleZoo, 29

getBoolProperty
    ParticleZoo::Particle, 130
getByteOrder
    ParticleZoo::ByteBuffer, 40
    ParticleZoo::IAEAphspFile::IAEAHeader, 84
    ParticleZoo::PhaseSpaceFileWriter, 189
getChecksum
    ParticleZoo::IAEAphspFile::IAEAHeader, 84
getCLICommands
    ParticleZoo::PhaseSpaceFileReader, 163
    ParticleZoo::PhaseSpaceFileWriter, 190
getColumnName
    ParticleZoo::TOPASphspFile::Header::DataColumn, 230
getColumnType
    ParticleZoo::TOPASphspFile::Header::DataColumn, 231

getColumnTypes
ParticleZoo::TOPASphspFile::Header, 223
getConstantPx
ParticleZoo::PhaseSpaceFileReader, 163
ParticleZoo::PhaseSpaceFileWriter, 190
getConstantPy
ParticleZoo::PhaseSpaceFileReader, 164
ParticleZoo::PhaseSpaceFileWriter, 190
getConstantPz
ParticleZoo::PhaseSpaceFileReader, 164
ParticleZoo::PhaseSpaceFileWriter, 191
getConstantU
ParticleZoo::IAEAphspFile::IAEAHeader, 84
getConstantV
ParticleZoo::IAEAphspFile::IAEAHeader, 85
getConstantW
ParticleZoo::IAEAphspFile::IAEAHeader, 85
getConstantWeight
ParticleZoo::IAEAphspFile::IAEAHeader, 85
ParticleZoo::PhaseSpaceFileReader, 165
ParticleZoo::PhaseSpaceFileWriter, 191
getConstantX
ParticleZoo::IAEAphspFile::IAEAHeader, 85
ParticleZoo::PhaseSpaceFileReader, 165
ParticleZoo::PhaseSpaceFileWriter, 192
getConstantY
ParticleZoo::IAEAphspFile::IAEAHeader, 86
ParticleZoo::PhaseSpaceFileReader, 165
ParticleZoo::PhaseSpaceFileWriter, 192
getConstantZ
ParticleZoo::IAEAphspFile::IAEAHeader, 86
ParticleZoo::PhaseSpaceFileReader, 166
ParticleZoo::PhaseSpaceFileWriter, 192
getCustomBoolProperties
ParticleZoo::Particle, 131
getCustomFloatProperties
ParticleZoo::Particle, 131
getCustomIntProperties
ParticleZoo::Particle, 131
getCustomStringProperties
ParticleZoo::Particle, 131
getDataFilePath
ParticleZoo::IAEAphspFile::IAEAHeader, 86
getDataType
ParticleZoo::TOPASphspFile::Header::DataColumn, 231
getDirectionalCosineX
ParticleZoo::Particle, 132
getDirectionalCosineY
ParticleZoo::Particle, 132
getDirectionalCosineZ
ParticleZoo::Particle, 132
getExtraFloatType
ParticleZoo::IAEAphspFile::IAEAHeader, 86
getExtraLongType
ParticleZoo::IAEAphspFile::IAEAHeader, 87
getFileName
ParticleZoo::PhaseSpaceFileReader, 166
ParticleZoo::PhaseSpaceFileWriter, 193
getFileSize
ParticleZoo::PhaseSpaceFileReader, 166
getFileType
ParticleZoo::IAEAphspFile::IAEAHeader, 87
getFixedValues
ParticleZoo::PhaseSpaceFileReader, 166
ParticleZoo::PhaseSpaceFileWriter, 193
getFloatProperty
ParticleZoo::Particle, 132
getFormatSpecificCLICommands
ParticleZoo::EGSphspFile::Reader, 52
ParticleZoo::EGSphspFile::Writer, 60
ParticleZoo::IAEAphspFile::Reader, 113
ParticleZoo::IAEAphspFile::Writer, 124
ParticleZoo::ROOT::Reader, 210
ParticleZoo::ROOT::Writer, 216
ParticleZoo::TOPASphspFile::Reader, 238
ParticleZoo::TOPASphspFile::Writer, 248
getHeader
ParticleZoo::IAEAphspFile::Reader, 113
ParticleZoo::IAEAphspFile::Writer, 124
ParticleZoo::TOPASphspFile::Reader, 239
ParticleZoo::TOPASphspFile::Writer, 248
getHeaderData
ParticleZoo::PhaseSpaceFileReader, 167
getHeaderFilePath
ParticleZoo::IAEAphspFile::IAEAHeader, 88
getHistoriesRead
ParticleZoo::PhaseSpaceFileReader, 167
getHistoriesWritten
ParticleZoo::PhaseSpaceFileWriter, 193
getIAEAIndex
ParticleZoo::IAEAphspFile::IAEAHeader, 88
getIncrementalHistories
ParticleZoo::Particle, 133
getIntProperty
ParticleZoo::Particle, 133

getKineticEnergy
ParticleZoo::Particle, 134
getMaxEnergy
ParticleZoo::IAEAphspFile::IAEAHeader, 88
getMaximumASCIILineLength
ParticleZoo::penEasyphspFile::Reader, 149
ParticleZoo::penEasyphspFile::Writer, 156
ParticleZoo::PhaseSpaceFileReader, 168
ParticleZoo::PhaseSpaceFileWriter, 193
ParticleZoo::TOPASphspFile::Reader, 239
ParticleZoo::TOPASphspFile::Writer, 248
getMaximumSupportedParticles
ParticleZoo::EGSphspFile::Writer, 60
ParticleZoo::IAEAphspFile::Writer, 124
ParticleZoo::penEasyphspFile::Writer, 156
ParticleZoo::PhaseSpaceFileWriter, 194
ParticleZoo::ROOT::Writer, 216
ParticleZoo::TOPASphspFile::Writer, 249
getMaxKineticEnergy
ParticleZoo::EGSphspFile::Reader, 52
getMaxKineticEnergyOfType
ParticleZoo::TOPASphspFile::Header, 224
getMaxWeight
ParticleZoo::IAEAphspFile::IAEAHeader, 88
getMaxX
ParticleZoo::IAEAphspFile::IAEAHeader, 89
getMaxY
ParticleZoo::IAEAphspFile::IAEAHeader, 89
getMaxZ
ParticleZoo::IAEAphspFile::IAEAHeader, 89
getMeanEnergy
ParticleZoo::IAEAphspFile::IAEAHeader, 90
getMeanWeight
ParticleZoo::IAEAphspFile::IAEAHeader, 90
getMinElectronEnergy
ParticleZoo::EGSphspFile::Reader, 53
getMinEnergy
ParticleZoo::IAEAphspFile::IAEAHeader, 91
getMinKineticEnergyOfType
ParticleZoo::TOPASphspFile::Header, 224
getMinWeight
ParticleZoo::IAEAphspFile::IAEAHeader, 91
getMinX
ParticleZoo::IAEAphspFile::IAEAHeader, 91
getMinY
ParticleZoo::IAEAphspFile::IAEAHeader, 92
getMinZ
ParticleZoo::IAEAphspFile::IAEAHeader, 92
getMode
ParticleZoo::EGSphspFile::Reader, 53
getNextParticle
ParticleZoo::PhaseSpaceFileReader, 168
getNumberOfBoolProperties
ParticleZoo::Particle, 134
getNumberOfEntriesInFile
ParticleZoo::PhaseSpaceFileReader, 169
getNumberOfExtraFloats
ParticleZoo::IAEAphspFile::IAEAHeader, 92
getNumberOfExtraLongs
ParticleZoo::IAEAphspFile::IAEAHeader, 92
getNumberOfFloatProperties
ParticleZoo::Particle, 134
getNumberOfIntProperties
ParticleZoo::Particle, 134
getNumberOfOriginalHistories
ParticleZoo::EGSphspFile::Reader, 53
ParticleZoo::IAEAphspFile::Reader, 114
ParticleZoo::penEasyphspFile::Reader, 149
ParticleZoo::PhaseSpaceFileReader, 169
ParticleZoo::ROOT::Reader, 210
ParticleZoo::TOPASphspFile::Header, 224
ParticleZoo::TOPASphspFile::Reader, 239
getNumberOfParticles
ParticleZoo::EGSphspFile::Reader, 53
ParticleZoo::IAEAphspFile::IAEAHeader, 93
ParticleZoo::IAEAphspFile::Reader, 114
ParticleZoo::penEasyphspFile::Reader, 149
ParticleZoo::PhaseSpaceFileReader, 169
ParticleZoo::ROOT::Reader, 210
ParticleZoo::TOPASphspFile::Header, 225
ParticleZoo::TOPASphspFile::Reader, 239
getNumberOfParticlesOfType
ParticleZoo::TOPASphspFile::Header, 225
getNumberOfPhotons
ParticleZoo::EGSphspFile::Reader, 54
getNumberOfRepresentedHistories
ParticleZoo::TOPASphspFile::Header, 225
getOriginalHistories
ParticleZoo::IAEAphspFile::IAEAHeader, 93
getParticleRecordLength
ParticleZoo::EGSphspFile::Reader, 54
ParticleZoo::EGSphspFile::Writer, 60
ParticleZoo::IAEAphspFile::Reader, 115
ParticleZoo::IAEAphspFile::Writer, 125
ParticleZoo::PhaseSpaceFileReader, 170
ParticleZoo::PhaseSpaceFileWriter, 194

ParticleZoo::TOPASphspFile::Reader, 240
ParticleZoo::TOPASphspFile::Writer, 249
getParticleRecordStartOffset
ParticleZoo::EGSphspFile::Reader, 54
ParticleZoo::EGSphspFile::Writer, 61
ParticleZoo::IAEAphspFile::Reader, 115
ParticleZoo::penEasyphspFile::Writer, 156
ParticleZoo::PhaseSpaceFileReader, 170
ParticleZoo::PhaseSpaceFileWriter, 195
getParticlesRead
ParticleZoo::PhaseSpaceFileReader, 171
getParticlesWritten
ParticleZoo::PhaseSpaceFileWriter, 195
getParticleTypeFromPDGID
ParticleZoo, 31
getParticleTypeName
ParticleZoo, 31
getPDGIDFromParticleType
ParticleZoo, 32
getPendingHistories
ParticleZoo::PhaseSpaceFileWriter, 195
ParticleZoo::TOPASphspFile::Writer, 249
getPHSPFormat
ParticleZoo::PhaseSpaceFileReader, 171
ParticleZoo::PhaseSpaceFileWriter, 195
getRecordLength
ParticleZoo::IAEAphspFile::IAEAHeader, 94
ParticleZoo::TOPASphspFile::Header, 225
getSection
ParticleZoo::IAEAphspFile::IAEAHeader, 94
getTitle
ParticleZoo::IAEAphspFile::IAEAHeader, 95
getTOPASFormat
ParticleZoo::TOPASphspFile::Header, 226
ParticleZoo::TOPASphspFile::Reader, 240
ParticleZoo::TOPASphspFile::Writer, 250
getTOPASFormatName
ParticleZoo::TOPASphspFile::Header, 226
getTotalWeight
ParticleZoo::IAEAphspFile::IAEAHeader, 95
getType
ParticleZoo::Particle, 135
getUserOptions
ParticleZoo::PhaseSpaceFileReader, 172
ParticleZoo::PhaseSpaceFileWriter, 196
GetVersionString
ParticleZoo::Version, 254
getWeight
ParticleZoo::Particle, 135
getX
ParticleZoo::Particle, 135
getY
ParticleZoo::Particle, 135
getZ
ParticleZoo::Particle, 136
GLOBAL_PARTICLE_ENERGY_CUTOFF
ParticleZoo::IAEAphspFile::IAEAHeader, 81
GLOBAL_PHOTON_ENERGY_CUTOFF
ParticleZoo::IAEAphspFile::IAEAHeader, 81

hasBoolProperty
ParticleZoo::Particle, 136
hasExtraFloat
ParticleZoo::IAEAphspFile::IAEAHeader, 95
hasExtraLong
ParticleZoo::IAEAphspFile::IAEAHeader, 96
hasFloatProperty
ParticleZoo::Particle, 136
hasIntProperty
ParticleZoo::Particle, 137
hasMoreParticles
ParticleZoo::PhaseSpaceFileReader, 172
Header
ParticleZoo::TOPASphspFile::Header, 222
HOST_BYTE_ORDER
ParticleZoo, 33

IAEA_INDEX
ParticleZoo::IAEAphspFile::IAEAHeader, 81
IAEAHeader
ParticleZoo::IAEAphspFile::IAEAHeader, 82
INCREMENTAL_HISTORY_NUMBER
ParticleZoo, 29
ParticleZoo::IAEAphspFile::IAEAHeader, 80
INITIAL_DIRECTION_COSINE_X
ParticleZoo::TOPASphspFile::Header, 221
INITIAL_DIRECTION_COSINE_Y
ParticleZoo::TOPASphspFile::Header, 221
INITIAL_DIRECTION_COSINE_Z
ParticleZoo::TOPASphspFile::Header, 221
INITIAL_KINETIC_ENERGY
ParticleZoo::TOPASphspFile::Header, 221
INITIAL_SOURCE_DESCRIPTION
ParticleZoo::IAEAphspFile::IAEAHeader, 81
INSTITUTION
ParticleZoo::IAEAphspFile::IAEAHeader, 81

INT32
    ParticleZoo::TOPASphspFile::Header, 222
INT8
    ParticleZoo::TOPASphspFile::Header, 222
IntPropertyType
    ParticleZoo, 29
INVALID
    ParticleZoo, 28, 29
IS_MULTIPLE_CROSSER
    ParticleZoo, 28
IS_SECONDARY_PARTICLE
    ParticleZoo, 28
isNewHistory
    ParticleZoo::Particle, 137
isPxConstant
    ParticleZoo::PhaseSpaceFileReader, 172
    ParticleZoo::PhaseSpaceFileWriter, 196
isPyConstant
    ParticleZoo::PhaseSpaceFileReader, 172
    ParticleZoo::PhaseSpaceFileWriter, 196
isPzConstant
    ParticleZoo::PhaseSpaceFileReader, 173
    ParticleZoo::PhaseSpaceFileWriter, 196
isWeightConstant
    ParticleZoo::PhaseSpaceFileReader, 173
    ParticleZoo::PhaseSpaceFileWriter, 197
isXConstant
    ParticleZoo::PhaseSpaceFileReader, 173
    ParticleZoo::PhaseSpaceFileWriter, 197
isYConstant
    ParticleZoo::PhaseSpaceFileReader, 173
    ParticleZoo::PhaseSpaceFileWriter, 197
isZConstant
    ParticleZoo::PhaseSpaceFileReader, 174
    ParticleZoo::PhaseSpaceFileWriter, 197

length
    ParticleZoo::ByteBuffer, 40
License, 17
LINK_VALIDATION
    ParticleZoo::IAEAphspFile::IAEAHeader, 81
LittleEndian
    ParticleZoo, 28

MACHINE_TYPE
    ParticleZoo::IAEAphspFile::IAEAHeader, 81
MAJOR_VERSION
    ParticleZoo::Version, 254
MC_INPUT_FILENAME
    ParticleZoo::IAEAphspFile::IAEAHeader, 81
MINOR_VERSION
    ParticleZoo::Version, 254
MONTE_CARLO_CODE_VERSION
    ParticleZoo::IAEAphspFile::IAEAHeader, 81
moveTo
    ParticleZoo::ByteBuffer, 41
moveToParticle
    ParticleZoo::PhaseSpaceFileReader, 174

NEUTRONS
    ParticleZoo::IAEAphspFile::IAEAHeader, 81
NEW_HISTORY_FLAG
    ParticleZoo::TOPASphspFile::Header, 221
NOMINAL_SSD
    ParticleZoo::IAEAphspFile::IAEAHeader, 81
NONE
    ParticleZoo, 29

operator<<
    ParticleZoo, 32
operator==
    ParticleZoo::FixedValues, 65
ORIGINAL_HISTORIES
    ParticleZoo::IAEAphspFile::IAEAHeader, 81

PARENT_ID
    ParticleZoo::TOPASphspFile::Header, 221
Particle
    ParticleZoo::Particle, 129
PARTICLE_TYPE
    ParticleZoo::TOPASphspFile::Header, 221
PARTICLES
    ParticleZoo::IAEAphspFile::IAEAHeader, 81
ParticleType
    ParticleZoo, 30
ParticleZoo, 1, 26
    ASCII, 29
    BigEndian, 28
    BINARY, 29
    BoolPropertyType, 28
    ByteOrder, 28
    CUSTOM, 28–30
    EGS_LATCH, 29
    FloatPropertyType, 28
    FormatType, 29
    getParticleTypeFromPDGID, 31

getParticleTypeName, 31
getPDGIDFromParticleType, 32
HOST_BYTE_ORDER, 33
INCREMENTAL_HISTORY_NUMBER, 29
IntPropertyType, 29
INVALID, 28, 29
IS_MULTIPLE_CROSSER, 28
IS_SECONDARY_PARTICLE, 28
LittleEndian, 28
NONE, 29
operator<<, 32
ParticleType, 30
PDPEndian, 28
PENELOPE_ILB1, 29
PENELOPE_ILB2, 30
PENELOPE_ILB3, 30
PENELOPE_ILB4, 30
PENELOPE_ILB5, 30
PseudoParticle, 30
Unsupported, 30
XLAST, 29
YLAST, 29
ZLAST, 29
ParticleZoo::ByteBuffer, 36
appendData, 38, 39
ByteBuffer, 37, 38
capacity, 39
compact, 39
data, 40
expand, 40
getByteOrder, 40
length, 40
moveTo, 41
read, 41
readBytes, 42
readLine, 42
readString, 43
remainingToRead, 44
remainingToWrite, 44
setByteOrder, 44
setData, 45
write, 46
writeBytes, 46
writeString, 47
ParticleZoo::EGSphspFile::Reader, 48
getFormatSpecificCLICommands, 52
getMaxKineticEnergy, 52
getMinElectronEnergy, 53
getMode, 53
getNumberOfOriginalHistories, 53
getNumberOfParticles, 53
getNumberOfPhotons, 54
getParticleRecordLength, 54
getParticleRecordStartOffset, 54
readBinaryParticle, 55
Reader, 52
ParticleZoo::EGSphspFile::Writer, 56
getFormatSpecificCLICommands, 60
getMaximumSupportedParticles, 60
getParticleRecordLength, 60
getParticleRecordStartOffset, 61
setNumberOfOriginalHistories, 61
writeBinaryParticle, 61
writeHeaderData, 62
Writer, 60
ParticleZoo::FixedValues, 64
operator==, 65
ParticleZoo::FormatRegistry, 66
CreateReader, 68
CreateWriter, 69, 70
ExtensionForFormat, 70
FILE_EXTENSION_CAN_HAVE_SUFFIX, 73
FormatsForExtension, 71
PrintSupportedFormats, 71
ReaderFactoryFn, 67
RegisterFormat, 71
RegisterStandardFormats, 72
SupportedFormats, 72
WriterFactoryFn, 67
ParticleZoo::IAEAphspFile::IAEAHeader, 74
addExtraFloat, 83
addExtraLong, 83
ADDITIONAL_NOTES, 81
AUTHORS, 81
BEAM_NAME, 81
BYTE_ORDER, 81
CHECKSUM, 81
checksumIsValid, 83
COORDINATE_SYSTEM_DESCRIPTION, 81
countParticleStats, 83
CUSTOM_SECTION, 81
DeterminePathToHeaderFile, 84
EGS_LATCH, 80
ELECTRONS, 81
EXTRA_FLOAT_TYPE, 79
EXTRA_LONG_TYPE, 79

FIELD_SIZE, 81
FILE_TYPE, 81
FileType, 80
getByteOrder, 84
getChecksum, 84
getConstantU, 84
getConstantV, 85
getConstantW, 85
getConstantWeight, 85
getConstantX, 85
getConstantY, 86
getConstantZ, 86
getDataFilePath, 86
getExtraFloatType, 86
getExtraLongType, 87
getFileType, 87
getHeaderFilePath, 88
getIAEAIndex, 88
getMaxEnergy, 88
getMaxWeight, 88
getMaxX, 89
getMaxY, 89
getMaxZ, 89
getMeanEnergy, 90
getMeanWeight, 90
getMinEnergy, 91
getMinWeight, 91
getMinX, 91
getMinY, 92
getMinZ, 92
getNumberOfExtraFloats, 92
getNumberOfExtraLongs, 92
getNumberOfParticles, 93
getOriginalHistories, 93
getRecordLength, 94
getSection, 94
getTitle, 95
getTotalWeight, 95
GLOBAL_PARTICLE_ENERGY_CUTOFF, 81
GLOBAL_PHOTON_ENERGY_CUTOFF, 81
hasExtraFloat, 95
hasExtraLong, 96
IAEA_INDEX, 81
IAEAHeader, 82
INCREMENTAL_HISTORY_NUMBER, 80
INITIAL_SOURCE_DESCRIPTION, 81
INSTITUTION, 81
LINK_VALIDATION, 81
MACHINE_TYPE, 81
MC_INPUT_FILENAME, 81
MONTE_CARLO_CODE_VERSION, 81
NEUTRONS, 81
NOMINAL_SSD, 81
ORIGINAL_HISTORIES, 81
PARTICLES, 81
PENELOPE_ILB1, 80
PENELOPE_ILB2, 80
PENELOPE_ILB3, 80
PENELOPE_ILB4, 80
PENELOPE_ILB5, 80
PHOTONS, 81
PHSP_FILE, 80
PHSP_GENERATOR, 80
POSITRONS, 81
PROTONS, 81
PUBLISHED_REFERENCE, 81
RECORD_CONSTANT, 81
RECORD_CONTENTS, 81
RECORD_LENGTH, 81
SECTION, 80
setChecksum, 96
setConstantU, 96
setConstantV, 97
setConstantW, 97
setConstantWeight, 97
setConstantX, 97
setConstantY, 98
setConstantZ, 98
setFilePath, 98
setFileType, 99
setIAEAIndex, 99
setMaxEnergy, 99
setMaxWeight, 100
setMaxX, 100
setMaxY, 100
setMaxZ, 100
setMeanEnergy, 101
setMinEnergy, 101
setMinWeight, 101
setMinX, 102
setMinY, 102
setMinZ, 102
setNumberOfParticles, 103
setOriginalHistories, 103
setRecordLength, 104
setSection, 104

setTitle, 105
setTotalWeight, 105
STATISTICAL_INFORMATION_GEOMETRY, 81
STATISTICAL_INFORMATION_PARTICLES, 81
TITLE, 81
translateExtraFloatType, 105
translateExtraLongType, 106
TRANSPORT_PARAMETERS, 81
uIsStored, 106
USER_DEFINED_GENERIC_TYPE, 79, 80
VARIANCE_REDUCTION_TECHNIQUES, 81
vIsStored, 106
weightIsStored, 107
wIsStored, 107
writeHeader, 107
xIsStored, 108
XLAST, 79
yIsStored, 108
YLAST, 79
zIsStored, 108
ZLAST, 79

ParticleZoo::IAEAphspFile::Reader, 109
getFormatSpecificCLICommands, 113
getHeader, 113
getNumberOfOriginalHistories, 114
getNumberOfParticles, 114
getParticleRecordLength, 115
getParticleRecordStartOffset, 115
readBinaryParticle, 115
Reader, 113

ParticleZoo::IAEAphspFile::Writer, 117
canHaveConstantPx, 122
canHaveConstantPy, 122
canHaveConstantPz, 122
canHaveConstantWeight, 122
canHaveConstantX, 123
canHaveConstantY, 123
canHaveConstantZ, 123
fixedValuesHaveChanged, 124
getFormatSpecificCLICommands, 124
getHeader, 124
getMaximumSupportedParticles, 124
getParticleRecordLength, 125
setNumberOfOriginalHistories, 125
writeBinaryParticle, 125
writeHeaderData, 126
Writer, 121

ParticleZoo::Particle, 127
getBoolProperty, 130
getCustomBoolProperties, 131
getCustomFloatProperties, 131
getCustomIntProperties, 131
getCustomStringProperties, 131
getDirectionalCosineX, 132
getDirectionalCosineY, 132
getDirectionalCosineZ, 132
getFloatProperty, 132
getIncrementalHistories, 133
getIntProperty, 133
getKineticEnergy, 134
getNumberOfBoolProperties, 134
getNumberOfFloatProperties, 134
getNumberOfIntProperties, 134
getType, 135
getWeight, 135
getX, 135
getY, 135
getZ, 136
hasBoolProperty, 136
hasFloatProperty, 136
hasIntProperty, 137
isNewHistory, 137
Particle, 129
projectToXValue, 137
projectToYValue, 138
projectToZValue, 138
reserveBoolProperties, 139
reserveFloatProperties, 139
reserveIntProperties, 140
setBoolProperty, 140
setDirectionalCosineX, 140
setDirectionalCosineY, 140
setDirectionalCosineZ, 141
setFloatProperty, 141
setIncrementalHistories, 141
setIntProperty, 142
setKineticEnergy, 142
setNewHistory, 142
setStringProperty, 143
setWeight, 143
setX, 143
setY, 144
setZ, 144

ParticleZoo::penEasyphspFile::Reader, 145
getMaximumASCIILineLength, 149
getNumberOfOriginalHistories, 149

getNumberOfParticles, 149
readASCIIParticle, 150
Reader, 148
ParticleZoo::penEasyphspFile::Writer, 152
getMaximumASCIILineLength, 156
getMaximumSupportedParticles, 156
getParticleRecordStartOffset, 156
writeASCIIParticle, 156
writeHeaderData, 157
Writer, 155
ParticleZoo::PhaseSpaceFileReader, 158
∼PhaseSpaceFileReader, 162
calcThirdUnitComponent, 162
close, 163
getCLICommands, 163
getConstantPx, 163
getConstantPy, 164
getConstantPz, 164
getConstantWeight, 165
getConstantX, 165
getConstantY, 165
getConstantZ, 166
getFileName, 166
getFileSize, 166
getFixedValues, 166
getHeaderData, 167
getHistoriesRead, 167
getMaximumASCIILineLength, 168
getNextParticle, 168
getNumberOfEntriesInFile, 169
getNumberOfOriginalHistories, 169
getNumberOfParticles, 169
getParticleRecordLength, 170
getParticleRecordStartOffset, 170
getParticlesRead, 171
getPHSPFormat, 171
getUserOptions, 172
hasMoreParticles, 172
isPxConstant, 172
isPyConstant, 172
isPzConstant, 173
isWeightConstant, 173
isXConstant, 173
isYConstant, 173
isZConstant, 174
moveToParticle, 174
PhaseSpaceFileReader, 161
readASCIIParticle, 174
readBinaryParticle, 175
readParticleManually, 176
setByteOrder, 176
setCommentMarkers, 176
setConstantPx, 177
setConstantPy, 177
setConstantPz, 177
setConstantWeight, 178
setConstantX, 178
setConstantY, 178
setConstantZ, 180
ParticleZoo::PhaseSpaceFileWriter, 181
∼PhaseSpaceFileWriter, 185
accountForAdditionalHistories, 185
addAdditionalHistories, 186
canHaveConstantPx, 186
canHaveConstantPy, 186
canHaveConstantPz, 187
canHaveConstantWeight, 187
canHaveConstantX, 187
canHaveConstantY, 188
canHaveConstantZ, 188
canWritePseudoParticlesExplicitly, 188
close, 189
fixedValuesHaveChanged, 189
getByteOrder, 189
getCLICommands, 190
getConstantPx, 190
getConstantPy, 190
getConstantPz, 191
getConstantWeight, 191
getConstantX, 192
getConstantY, 192
getConstantZ, 192
getFileName, 193
getFixedValues, 193
getHistoriesWritten, 193
getMaximumASCIILineLength, 193
getMaximumSupportedParticles, 194
getParticleRecordLength, 194
getParticleRecordStartOffset, 195
getParticlesWritten, 195
getPendingHistories, 195
getPHSPFormat, 195
getUserOptions, 196
isPxConstant, 196
isPyConstant, 196
isPzConstant, 196

isWeightConstant, 197
isXConstant, 197
isYConstant, 197
isZConstant, 197
PhaseSpaceFileWriter, 184
setByteOrder, 198
setConstantPx, 198
setConstantPy, 198
setConstantPz, 200
setConstantWeight, 200
setConstantX, 200
setConstantY, 200
setConstantZ, 201
writeASCIIParticle, 201
writeBinaryParticle, 202
writeHeaderData, 202
writeParticle, 203
writeParticleManually, 203
ParticleZoo::ROOT::BranchInfo, 205
ParticleZoo::ROOT::Reader, 206
getFormatSpecificCLICommands, 210
getNumberOfOriginalHistories, 210
getNumberOfParticles, 210
Reader, 210
readParticleManually, 211
ParticleZoo::ROOT::Writer, 212
getFormatSpecificCLICommands, 216
getMaximumSupportedParticles, 216
writeHeaderData, 216
writeParticleManually, 217
Writer, 216
ParticleZoo::SupportedFormat, 218
ParticleZoo::TOPASphspFile::Header, 219
addColumnType, 223
BOOLEAN, 222
CHARGE, 221
ColumnType, 221
countParticleStats, 223
CREATOR_PROCESS, 221
DataType, 222
DIRECTION_COSINE_X, 221
DIRECTION_COSINE_Y, 221
DIRECTION_COSINE_Z_SIGN, 221
ENERGY, 221
EVENT_ID, 221
FLOAT32, 222
FLOAT64, 222
getColumnTypes, 223
getMaxKineticEnergyOfType, 224
getMinKineticEnergyOfType, 224
getNumberOfOriginalHistories, 224
getNumberOfParticles, 225
getNumberOfParticlesOfType, 225
getNumberOfRepresentedHistories, 225
getRecordLength, 225
getTOPASFormat, 226
getTOPASFormatName, 226
Header, 222
INITIAL_DIRECTION_COSINE_X, 221
INITIAL_DIRECTION_COSINE_Y, 221
INITIAL_DIRECTION_COSINE_Z, 221
INITIAL_KINETIC_ENERGY, 221
INT32, 222
INT8, 222
NEW_HISTORY_FLAG, 221
PARENT_ID, 221
PARTICLE_TYPE, 221
POSITION_X, 221
POSITION_Y, 221
POSITION_Z, 221
RUN_ID, 221
SEED_PART_1, 221
SEED_PART_2, 221
SEED_PART_3, 221
SEED_PART_4, 221
setNumberOfOriginalHistories, 227
STRING, 222
TIME_OF_FLIGHT, 221
TOPAS_TIME, 221
TRACK_ID, 221
VERTEX_POSITION_X, 221
VERTEX_POSITION_Y, 221
VERTEX_POSITION_Z, 221
WEIGHT, 221
writeHeader, 227
ParticleZoo::TOPASphspFile::Header::DataColumn, 228
DataColumn, 229, 230
getColumnName, 230
getColumnType, 231
getDataType, 231
sizeOf, 232
ParticleZoo::TOPASphspFile::Header::ParticleStats, 233
ParticleZoo::TOPASphspFile::Reader, 234
getFormatSpecificCLICommands, 238
getHeader, 239
getMaximumASCIILineLength, 239

getNumberOfOriginalHistories, 239
getNumberOfParticles, 239
getParticleRecordLength, 240
getTOPASFormat, 240
readASCIIParticle, 240
readBinaryParticle, 241
Reader, 238
setDetailedReading, 241
ParticleZoo::TOPASphspFile::Writer, 243
accountForAdditionalHistories, 247
canWritePseudoParticlesExplicitly, 248
getFormatSpecificCLICommands, 248
getHeader, 248
getMaximumASCIILineLength, 248
getMaximumSupportedParticles, 249
getParticleRecordLength, 249
getPendingHistories, 249
getTOPASFormat, 250
writeASCIIParticle, 250
writeBinaryParticle, 251
writeHeaderData, 251
Writer, 247
ParticleZoo::Version, 253
CAVEAT, 254
GetVersionString, 254
MAJOR_VERSION, 254
MINOR_VERSION, 254
PATCH_VERSION, 254
PROJECT_NAME, 255
PATCH_VERSION
ParticleZoo::Version, 254
PDPEndian
ParticleZoo, 28
PENELOPE_ILB1
ParticleZoo, 29
ParticleZoo::IAEAphspFile::IAEAHeader, 80
PENELOPE_ILB2
ParticleZoo, 30
ParticleZoo::IAEAphspFile::IAEAHeader, 80
PENELOPE_ILB3
ParticleZoo, 30
ParticleZoo::IAEAphspFile::IAEAHeader, 80
PENELOPE_ILB4
ParticleZoo, 30
ParticleZoo::IAEAphspFile::IAEAHeader, 80
PENELOPE_ILB5
ParticleZoo, 30
ParticleZoo::IAEAphspFile::IAEAHeader, 80
PhaseSpaceFileReader
ParticleZoo::PhaseSpaceFileReader, 161
PhaseSpaceFileWriter
ParticleZoo::PhaseSpaceFileWriter, 184
PHOTONS
ParticleZoo::IAEAphspFile::IAEAHeader, 81
PHSP_FILE
ParticleZoo::IAEAphspFile::IAEAHeader, 80
PHSP_GENERATOR
ParticleZoo::IAEAphspFile::IAEAHeader, 80
POSITION_X
ParticleZoo::TOPASphspFile::Header, 221
POSITION_Y
ParticleZoo::TOPASphspFile::Header, 221
POSITION_Z
ParticleZoo::TOPASphspFile::Header, 221
POSITRONS
ParticleZoo::IAEAphspFile::IAEAHeader, 81
PrintSupportedFormats
ParticleZoo::FormatRegistry, 71
PROJECT_NAME
ParticleZoo::Version, 255
projectToXValue
ParticleZoo::Particle, 137
projectToYValue
ParticleZoo::Particle, 138
projectToZValue
ParticleZoo::Particle, 138
PROTONS
ParticleZoo::IAEAphspFile::IAEAHeader, 81
PseudoParticle
ParticleZoo, 30
PUBLISHED_REFERENCE
ParticleZoo::IAEAphspFile::IAEAHeader, 81

read
ParticleZoo::ByteBuffer, 41
readASCIIParticle
ParticleZoo::penEasyphspFile::Reader, 150
ParticleZoo::PhaseSpaceFileReader, 174
ParticleZoo::TOPASphspFile::Reader, 240
readBinaryParticle
ParticleZoo::EGSphspFile::Reader, 55
ParticleZoo::IAEAphspFile::Reader, 115
ParticleZoo::PhaseSpaceFileReader, 175
ParticleZoo::TOPASphspFile::Reader, 241
readBytes
ParticleZoo::ByteBuffer, 42

Reader
ParticleZoo::EGSphspFile::Reader, 52
ParticleZoo::IAEAphspFile::Reader, 113
ParticleZoo::penEasyphspFile::Reader, 148
ParticleZoo::ROOT::Reader, 210
ParticleZoo::TOPASphspFile::Reader, 238
ReaderFactoryFn
ParticleZoo::FormatRegistry, 67
readLine
ParticleZoo::ByteBuffer, 42
readParticleManually
ParticleZoo::PhaseSpaceFileReader, 176
ParticleZoo::ROOT::Reader, 211
readString
ParticleZoo::ByteBuffer, 43
RECORD_CONSTANT
ParticleZoo::IAEAphspFile::IAEAHeader, 81
RECORD_CONTENTS
ParticleZoo::IAEAphspFile::IAEAHeader, 81
RECORD_LENGTH
ParticleZoo::IAEAphspFile::IAEAHeader, 81
RegisterFormat
ParticleZoo::FormatRegistry, 71
RegisterStandardFormats
ParticleZoo::FormatRegistry, 72
remainingToRead
ParticleZoo::ByteBuffer, 44
remainingToWrite
ParticleZoo::ByteBuffer, 44
reserveBoolProperties
ParticleZoo::Particle, 139
reserveFloatProperties
ParticleZoo::Particle, 139
reserveIntProperties
ParticleZoo::Particle, 140
RUN_ID
ParticleZoo::TOPASphspFile::Header, 221

SECTION
ParticleZoo::IAEAphspFile::IAEAHeader, 80
SEED_PART_1
ParticleZoo::TOPASphspFile::Header, 221
SEED_PART_2
ParticleZoo::TOPASphspFile::Header, 221
SEED_PART_3
ParticleZoo::TOPASphspFile::Header, 221
SEED_PART_4
ParticleZoo::TOPASphspFile::Header, 221
setBoolProperty
ParticleZoo::Particle, 140
setByteOrder
ParticleZoo::ByteBuffer, 44
ParticleZoo::PhaseSpaceFileReader, 176
ParticleZoo::PhaseSpaceFileWriter, 198
setChecksum
ParticleZoo::IAEAphspFile::IAEAHeader, 96
setCommentMarkers
ParticleZoo::PhaseSpaceFileReader, 176
setConstantPx
ParticleZoo::PhaseSpaceFileReader, 177
ParticleZoo::PhaseSpaceFileWriter, 198
setConstantPy
ParticleZoo::PhaseSpaceFileReader, 177
ParticleZoo::PhaseSpaceFileWriter, 198
setConstantPz
ParticleZoo::PhaseSpaceFileReader, 177
ParticleZoo::PhaseSpaceFileWriter, 200
setConstantU
ParticleZoo::IAEAphspFile::IAEAHeader, 96
setConstantV
ParticleZoo::IAEAphspFile::IAEAHeader, 97
setConstantW
ParticleZoo::IAEAphspFile::IAEAHeader, 97
setConstantWeight
ParticleZoo::IAEAphspFile::IAEAHeader, 97
ParticleZoo::PhaseSpaceFileReader, 178
ParticleZoo::PhaseSpaceFileWriter, 200
setConstantX
ParticleZoo::IAEAphspFile::IAEAHeader, 97
ParticleZoo::PhaseSpaceFileReader, 178
ParticleZoo::PhaseSpaceFileWriter, 200
setConstantY
ParticleZoo::IAEAphspFile::IAEAHeader, 98
ParticleZoo::PhaseSpaceFileReader, 178
ParticleZoo::PhaseSpaceFileWriter, 200
setConstantZ
ParticleZoo::IAEAphspFile::IAEAHeader, 98
ParticleZoo::PhaseSpaceFileReader, 180
ParticleZoo::PhaseSpaceFileWriter, 201
setData
ParticleZoo::ByteBuffer, 45
setDetailedReading
ParticleZoo::TOPASphspFile::Reader, 241
setDirectionalCosineX
ParticleZoo::Particle, 140
setDirectionalCosineY

ParticleZoo::Particle, 140
setDirectionalCosineZ
ParticleZoo::Particle, 141
setFilePath
ParticleZoo::IAEAphspFile::IAEAHeader, 98
setFileType
ParticleZoo::IAEAphspFile::IAEAHeader, 99
setFloatProperty
ParticleZoo::Particle, 141
setIAEAIndex
ParticleZoo::IAEAphspFile::IAEAHeader, 99
setIncrementalHistories
ParticleZoo::Particle, 141
setIntProperty
ParticleZoo::Particle, 142
setKineticEnergy
ParticleZoo::Particle, 142
setMaxEnergy
ParticleZoo::IAEAphspFile::IAEAHeader, 99
setMaxWeight
ParticleZoo::IAEAphspFile::IAEAHeader, 100
setMaxX
ParticleZoo::IAEAphspFile::IAEAHeader, 100
setMaxY
ParticleZoo::IAEAphspFile::IAEAHeader, 100
setMaxZ
ParticleZoo::IAEAphspFile::IAEAHeader, 100
setMeanEnergy
ParticleZoo::IAEAphspFile::IAEAHeader, 101
setMinEnergy
ParticleZoo::IAEAphspFile::IAEAHeader, 101
setMinWeight
ParticleZoo::IAEAphspFile::IAEAHeader, 101
setMinX
ParticleZoo::IAEAphspFile::IAEAHeader, 102
setMinY
ParticleZoo::IAEAphspFile::IAEAHeader, 102
setMinZ
ParticleZoo::IAEAphspFile::IAEAHeader, 102
setNewHistory
ParticleZoo::Particle, 142
setNumberOfOriginalHistories
ParticleZoo::EGSphspFile::Writer, 61
ParticleZoo::IAEAphspFile::Writer, 125
ParticleZoo::TOPASphspFile::Header, 227
setNumberOfParticles
ParticleZoo::IAEAphspFile::IAEAHeader, 103
setOriginalHistories
ParticleZoo::IAEAphspFile::IAEAHeader, 103
setRecordLength
ParticleZoo::IAEAphspFile::IAEAHeader, 104
setSection
ParticleZoo::IAEAphspFile::IAEAHeader, 104
setStringProperty
ParticleZoo::Particle, 143
setTitle
ParticleZoo::IAEAphspFile::IAEAHeader, 105
setTotalWeight
ParticleZoo::IAEAphspFile::IAEAHeader, 105
setWeight
ParticleZoo::Particle, 143
setX
ParticleZoo::Particle, 143
setY
ParticleZoo::Particle, 144
setZ
ParticleZoo::Particle, 144
sizeOf
ParticleZoo::TOPASphspFile::Header::DataColumn, 232
STATISTICAL_INFORMATION_GEOMETRY
ParticleZoo::IAEAphspFile::IAEAHeader, 81
STATISTICAL_INFORMATION_PARTICLES
ParticleZoo::IAEAphspFile::IAEAHeader, 81
STRING
ParticleZoo::TOPASphspFile::Header, 222
SupportedFormats
ParticleZoo::FormatRegistry, 72

TIME_OF_FLIGHT
ParticleZoo::TOPASphspFile::Header, 221
TITLE
ParticleZoo::IAEAphspFile::IAEAHeader, 81
TOPAS_TIME
ParticleZoo::TOPASphspFile::Header, 221
TRACK_ID
ParticleZoo::TOPASphspFile::Header, 221
translateExtraFloatType
ParticleZoo::IAEAphspFile::IAEAHeader, 105
translateExtraLongType
ParticleZoo::IAEAphspFile::IAEAHeader, 106
TRANSPORT_PARAMETERS
ParticleZoo::IAEAphspFile::IAEAHeader, 81

uIsStored
ParticleZoo::IAEAphspFile::IAEAHeader, 106

Unsupported
  ParticleZoo, 30
USER_DEFINED_GENERIC_TYPE
  ParticleZoo::IAEAphspFile::IAEAHeader, 79, 80

VARIANCE_REDUCTION_TECHNIQUES
  ParticleZoo::IAEAphspFile::IAEAHeader, 81
VERTEX_POSITION_X
  ParticleZoo::TOPASphspFile::Header, 221
VERTEX_POSITION_Y
  ParticleZoo::TOPASphspFile::Header, 221
VERTEX_POSITION_Z
  ParticleZoo::TOPASphspFile::Header, 221
vIsStored
  ParticleZoo::IAEAphspFile::IAEAHeader, 106

WEIGHT
  ParticleZoo::TOPASphspFile::Header, 221
weightIsStored
  ParticleZoo::IAEAphspFile::IAEAHeader, 107
wIsStored
  ParticleZoo::IAEAphspFile::IAEAHeader, 107
write
  ParticleZoo::ByteBuffer, 46
writeASCIIParticle
  ParticleZoo::penEasyphspFile::Writer, 156
  ParticleZoo::PhaseSpaceFileWriter, 201
  ParticleZoo::TOPASphspFile::Writer, 250
writeBinaryParticle
  ParticleZoo::EGSphspFile::Writer, 61
  ParticleZoo::IAEAphspFile::Writer, 125
  ParticleZoo::PhaseSpaceFileWriter, 202
  ParticleZoo::TOPASphspFile::Writer, 251
writeBytes
  ParticleZoo::ByteBuffer, 46
writeHeader
  ParticleZoo::IAEAphspFile::IAEAHeader, 107
  ParticleZoo::TOPASphspFile::Header, 227
writeHeaderData
  ParticleZoo::EGSphspFile::Writer, 62
  ParticleZoo::IAEAphspFile::Writer, 126
  ParticleZoo::penEasyphspFile::Writer, 157
  ParticleZoo::PhaseSpaceFileWriter, 202
  ParticleZoo::ROOT::Writer, 216
  ParticleZoo::TOPASphspFile::Writer, 251
writeParticle
  ParticleZoo::PhaseSpaceFileWriter, 203
writeParticleManually
  ParticleZoo::PhaseSpaceFileWriter, 203
  ParticleZoo::ROOT::Writer, 217
Writer
  ParticleZoo::EGSphspFile::Writer, 60
  ParticleZoo::IAEAphspFile::Writer, 121
  ParticleZoo::penEasyphspFile::Writer, 155
  ParticleZoo::ROOT::Writer, 216
  ParticleZoo::TOPASphspFile::Writer, 247
WriterFactoryFn
  ParticleZoo::FormatRegistry, 67
writeString
  ParticleZoo::ByteBuffer, 47

xIsStored
  ParticleZoo::IAEAphspFile::IAEAHeader, 108
XLAST
  ParticleZoo, 29
  ParticleZoo::IAEAphspFile::IAEAHeader, 79

yIsStored
  ParticleZoo::IAEAphspFile::IAEAHeader, 108
YLAST
  ParticleZoo, 29
  ParticleZoo::IAEAphspFile::IAEAHeader, 79

zIsStored
  ParticleZoo::IAEAphspFile::IAEAHeader, 108
ZLAST
  ParticleZoo, 29
  ParticleZoo::IAEAphspFile::IAEAHeader, 79