LCIO

"2.7.4"

Generated by Doxygen 1.8.6

Fri Dec 2 2016 11:44:41

Contents

Chapter 1

LCIO (v02-07-04)

A persistency framework for linear collider simulation studies.

1.1 Overview

This is the API documentation for the C++ version of LCIO. For general documetation on LCIO see the `homepage`. There you can also find the API documentation of the Java version.

1.2 Usage

A good starting point to browse this documentation is the `Namespace List`. As all data entities inherit from **EVENT::LCObject** (p. 123), the corresponding class documentation shows the complete hierarchy of all objects that can be stored with LCIO. The namespace **IO** (p. 27) contains the abstract interfaces for reading and writing data.

Chapter 2

Deprecated List

See Also

Member EVENT::SimCalorimeterHit::getNMCParticles (p. 207) **() const =0**
getNMCContributions()

See Also

Member EVENT::SimTrackerHit::getdEdx (p. 215) **() const =0**
getEDep()

Author

Class EVENT::TPCHit (p. 258)
gaede

Version

See Also

Member EVENT::TrackerHit::getdEdx (p. 273) **() const =0**
getEDep()

See Also

Member IMPL::SimCalorimeterHitImpl::getNMCParticles (p. 211) **() const**
getNMCContributions()

See Also

Member IMPL::SimTrackerHitImpl::getdEdx (p. 218) **() const**
getEDep()

See Also

Member IMPL::SimTrackerHitImpl::setdEdx (p. 218) **(float dEdX)**
setEDep()

See Also

Member IMPL::TrackerHitImpl::getdEdx (p. 276) **() const**
getEDep()

See Also

Member IMPL::TrackerHitPlaneImpl::getdEdx (p. 282) **() const**
getEDep()

See Also

Member IMPL::TrackerHitZCylinderImpl::getdEdx (p. 287) **() const**
getEDep()

Member UTIL::LCStdHepRdr::readEvent (p. 156) **()**
please use updateEvent()

Member UTIL::LCStdHepRdrNew::readEvent (p. 158) **()**
please use updateEvent()

Chapter 3

Namespace Index

3.1 Namespace List

Here is a list of all documented namespaces with brief descriptions:

Chapter 4

Hierarchical Index

4.1 Class Hierarchy

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

Chapter 5

Class Index

5.1 Class List

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

Chapter 6

Namespace Documentation

6.1 DATA Namespace Reference

The namespace **DATA** (p. 21) has been removed from LCIO all data entities are now defined in **EVENT** (p. 21).

6.1.1 Detailed Description

The namespace **DATA** (p. 21) has been removed from LCIO all data entities are now defined in **EVENT** (p. 21).

6.2 EVENT Namespace Reference

The namespace **EVENT** (p. 21) holds all base interfaces of the event entities.

Classes

- class **CalorimeterHit**
  The generic calorimeter hit - created from ***SimCalorimeterHit*** *(p. 205) or* ***RawCalorimeterHit*** *(p. 188).*
- class **Cluster**
  The ***LCIO*** *(p. 115) cluster.*
- class **LCCollection**
  The generic collection used in ***LCIO*** *(p. 115).*
- class **LCEvent**
  The main event interface.
- class **LCFlag**
  Helper class to create and interpret the 32-bit flag word in LCCollections.
- class **LCFloatVec**
  Float vector used for user extensions.
- class **LCGenericObject**
  Simple interface to store generic user data.
- class **LCIntVec**
  Int vector used for user extensions.
- class **LCIO**
  Global constants used in ***LCIO*** *(p. 115).*
- class **LCObject**
  The generic object that is held in an ***LCCollection*** *(p. 83).*
- class **LCParameters**

  Simple interface to store generic named parameters of type int, float and string.
- class **LCRelation**
  A single weighted relationship between two LCObjects.
- class **LCRunHeader**
  Interface for the run header.
- class **LCStrVec**
  String vector used for user extensions.
- class **MCParticle**
  The ***LCIO*** *(p. 115) Monte Carlo particle.*
- class **ParticleID**
  Persistent interface for ***LCIO*** *(p. 115)* ***ParticleIDs*** *(p. 185).*
- class **RawCalorimeterHit**
  The generic calorimeter hit for real data (or simulation thereof).
- class **ReconstructedParticle**
  The ***LCIO*** *(p. 115) reconstructedParticle.*
- class **SimCalorimeterHit**
  The generic simulated calorimeter hit.
- class **SimTrackerHit**
  A generic simulated tracker hit.
- class **TPCHit**
  Deprecated.
- class **Track**
  The ***LCIO*** *(p. 115) track class.*
- class **TrackerData**
  TrackerData *(p. 267) contains the corrected (calibrated) raw tracker data.*
- class **TrackerHit**
  A generic tracker hit to be used by pattern recognition.
- class **TrackerHitPlane**
  A tracker hit on a planar surface - the hit is stored as:
  x,y,z, u(theta, phi), v(theta,phi), du, dv
  where (x,y,z) is the position in global cartesian coordinates, u,v are the the meassurement directions (unit vectors, spanning the plane) and du,dv are the measurement errors along these directions.
- class **TrackerHitZCylinder**
  A tracker hit on a cylindrical surface that runs parallel to the z-axis - the hit is stored as:
  x,y,z, center(x,y), dRPhi, dz
  where (x,y,z) is the position in global cartesian coordinates, center(x,y) is the center of the cylinder and Rphi and z are the measurement errors.
- class **TrackerPulse**
  Tracker pulses as computed from ***TrackerData*** *(p. 267) objects or as directly measured by a specific subdetector.*
- class **TrackerRawData**
  Generic class for raw tracker data.
- class **TrackState**
  The ***LCIO*** *(p. 115)* ***TrackState*** *(p. 304) class.*
- class **Vertex**
  The ***Vertex*** *(p. 312) class for* ***LCIO*** *(p. 115).*
- class **Exception**
  Base exception class for ***LCIO*** *(p. 115) - all other exceptions extend this.*
- class **EventException**
  EventException *(p. 69) used for errors accessing the event data.*
- class **DataNotAvailableException**
  EventException *(p. 69) used for data not available.*
- class **ReadOnlyException**
  EventException *(p. 69) used for signaling a 'read only exception'.*

Typedefs

- typedef std::vector
  < **CalorimeterHit** ∗ > **CalorimeterHitVec**

  Vector of (pointers to) CalorimeterHits.
- typedef std::vector< **Cluster** ∗ > **ClusterVec**

  Vector of (pointers to) Clusters.
- typedef std::vector< **LCObject** ∗ > **LCObjectVec**

  Vector of (pointers to) LCObjects.
- typedef std::vector< **MCParticle** ∗ > **MCParticleVec**

  Vector of (pointers to) MCParticles.
- typedef std::vector< **ParticleID** ∗ > **ParticleIDVec**

  *Vector of (pointers to) **ParticleIDs** (p. 185).*
- typedef std::vector
  < **RawCalorimeterHit** ∗ > **RawCalorimeterHitVec**

  Vector of (pointers to) RawCalorimeterHits.
- typedef std::vector
  < **ReconstructedParticle** ∗ > **ReconstructedParticleVec**

  Vector of (pointers to) ReconstructedParticles.
- typedef std::vector< **Track** ∗ > **TrackVec**

  Vector of (pointers to) Tracks.
- typedef std::vector< **TrackerHit** ∗ > **TrackerHitVec**

  Vector of (pointers to) TrackerHits.
- typedef std::vector< **TrackState** ∗ > **TrackStateVec**

  Vector of (pointers to) TrackStates.
- typedef std::vector< **Vertex** ∗ > **VertexVec**

  Vector of (pointers to) Vertices.
- typedef std::vector< std::string > **StringVec**

  Vector of strings.
- typedef std::vector< float > **FloatVec**

  Vector of floats.
- typedef std::vector< double > **DoubleVec**

  Vector of doubles.
- typedef std::vector< int > **IntVec**

  Vector of ints.
- typedef std::vector< short > **ShortVec**

  Vector of shorts.
- typedef long long **long64**

  64 bit signed integer,e.g.to be used for timestamps
- typedef unsigned long long **ulong64**

  64 bit unsigned integer,e.g.to be used for masks

6.2.1 Detailed Description

The namespace **EVENT** (p. 21) holds all base interfaces of the event entities. Typedefs used in **LCIO** (p. 115).

Typedefs for std::vectors of basic types used in **LCIO** (p. 115).

6.2.2 Typedef Documentation

6.2.2.1 typedef std::vector<CalorimeterHit∗> EVENT::CalorimeterHitVec

Vector of (pointers to) CalorimeterHits.

6.2.2.2 typedef std::vector<Cluster∗> EVENT::ClusterVec

Vector of (pointers to) Clusters.

6.2.2.3 typedef std::vector< double > EVENT::DoubleVec

Vector of doubles.

6.2.2.4 typedef std::vector< float > EVENT::FloatVec

Vector of floats.

6.2.2.5 typedef std::vector< int > EVENT::IntVec

Vector of ints.

6.2.2.6 typedef std::vector<LCObject∗> EVENT::LCObjectVec

Vector of (pointers to) LCObjects.

6.2.2.7 typedef std::vector<MCParticle∗> EVENT::MCParticleVec

Vector of (pointers to) MCParticles.

6.2.2.8 typedef std::vector<ParticleID∗> EVENT::ParticleIDVec

Vector of (pointers to) **ParticleIDs** (p. 185).

6.2.2.9 typedef std::vector<RawCalorimeterHit∗> EVENT::RawCalorimeterHitVec

Vector of (pointers to) RawCalorimeterHits.

6.2.2.10 typedef std::vector<ReconstructedParticle∗> EVENT::ReconstructedParticleVec

Vector of (pointers to) ReconstructedParticles.

6.2.2.11 typedef std::vector< short > EVENT::ShortVec

Vector of shorts.

6.2.2.12 typedef std::vector< std::string > EVENT::StringVec

Vector of strings.

6.2.2.13 typedef std::vector<TrackerHit∗> EVENT::TrackerHitVec

Vector of (pointers to) TrackerHits.

6.2.2.14 typedef std::vector< TrackState∗ > EVENT::TrackStateVec

Vector of (pointers to) TrackStates.

6.2.2.15 typedef std::vector< Track∗ > EVENT::TrackVec

Vector of (pointers to) Tracks.

6.2.2.16 typedef std::vector< Vertex∗ > EVENT::VertexVec

Vector of (pointers to) Vertices.

6.3 IMPL Namespace Reference

The namespace **IMPL** (p. 25) holds the implementations of the base interfaces that are defined in **EVENT** (p. 21).

Classes

- class **AccessChecked**
  Controls access to objects.
- class **CalorimeterHitImpl**
  Implementation of the real data CalorimeterHit.
- class **ClusterImpl**
  Implementation of Cluster.
- class **LCCollectionVec**
  Implementation of the LCCollection using (inheriting from) an STL vector of LCObjects.
- class **LCEventImpl**
  Implementation of the main event class.
- class **LCFlagImpl**
  Implementation of helper class to create and interpret the 32-bit flag word in LCCollections.
- class **LCGenericObjectImpl**
  Default LCIO implementation of the interface to store generic user data.
- class **LCIOExceptionHandler**
  Sets the default error handling for unexpected and uncaught exceptions.
- class **LCParametersImpl**
  Implementation of Simple interface to store generic named parameters of type int, float and string.
- class **LCRelationImpl**
  Implementation of the LCRelation.
- class **LCRunHeaderImpl**
  Implementation of LCRunHeader.
- class **MCParticleImpl**
  Implementation of MCParticle.
- class **PIDSort**
  *Helper class to sort **ParticleIDs** (p. 185) wrt.*
- class **ParticleIDImpl**
  Implementation of ParticleID.
- class **RawCalorimeterHitImpl**
  Implementation of the real data RawCalorimeterHit.
- class **ReconstructedParticleImpl**

  Implementation of ReconstructedParticle.
- struct **MCParticleCont**
- class **SimCalorimeterHitImpl**
  Implementation of the generic SimCalorimeterHit.
- class **SimTrackerHitImpl**
  Implementation of SimTrackerHit.
- class **TPCHitImpl**
  Implementation of the real data TPCHit.
- class **TrackerDataImpl**
  Default implementation of TrackerData.
- class **TrackerHitImpl**
  Implementation of the generic tracker hit.
- class **TrackerHitPlaneImpl**
  Implementation of the planar tracker hit.
- class **TrackerHitZCylinderImpl**
  Implementation of the tracker hit on a cylindrical surface parallel to z.
- class **TrackerPulseImpl**
  Default implementation of TrackerPulse.
- class **TrackerRawDataImpl**
  Default implementation of TrackerRawData.
- class **TrackImpl**
  Implementation of the LCIO track class.
- class **TrackStateImpl**
  Implementation of the LCIO TrackState class.
- class **VertexImpl**
  Implementation of Vertex Class.

Typedefs

- typedef std::map< std::string, **EVENT::LCCollection** ∗ > **LCCollectionMap**
- typedef std::set < **EVENT::LCCollection** ∗ > **LCCollectionSet**
- typedef std::map< std::string, **EVENT::IntVec** > **IntMap**
- typedef std::map< std::string, **EVENT::FloatVec** > **FloatMap**
- typedef std::map< std::string, **EVENT::StringVec** > **StringMap**
- typedef std::vector < **IMPL::MCParticleCont** ∗ > **MCParticleContVec**

Functions

- void **lcio_unexpected** ()
  Method catches any std::exception, prints the message to stdout and then exits the program.

6.3.1 Detailed Description

The namespace **IMPL** (p. 25) holds the implementations of the base interfaces that are defined in **EVENT** (p. 21).

6.4 IO Namespace Reference

The namespace **IO** (p. 27) holds the base interfaces for io of data.

Classes

- class **ILCFactory**
  Factory to create ***LCWriter*** *(p. 166) and Reader object for the known data formats, e.g.*
- class **LCEventListener**
  Listener for the occurence of LCEvents when reading a stream.
- class **LCReader**
  Interface for reading data from LCIO.
- class **LCRunListener**
  Listener for the occurence of LCRunHeader when reading a stream.
- class **LCWriter**
  Interface for writing data with LCIO.
- class **IOException**
  IOException *(p. 79) used for reading/writing errors.*
- class **EndOfDataException**
  EndOfDataException *(p. 68) for signaling the end of a data stream.*

6.4.1 Detailed Description

The namespace **IO** (p. 27) holds the base interfaces for io of data.

6.5 IOIMPL Namespace Reference

The **IOIMPL** (p. 27) namespace holds extensions to the default implementations needed for **IO** (p. 27).

Classes

- class **CalorimeterHitIOImpl**
  Adding stuff needed for io (friend declarations, etc.)
- class **ClusterIOImpl**
  Adding stuff needed for io (friend declarations, etc.)
- class **LCCollectionIOVec**
  Adding stuff needed for io (friend declarations, etc.)
- class **LCEventIOImpl**
  Adding stuff needed for io (friend declarations, etc.)
- class **LCFactory**
  Factory to create LCWriter and Reader object for the known data formats, e.g.
- class **LCGenericObjectIOImpl**
  Adding stuff needed for io (friend declarations, etc.)
- class **LCRelationIOImpl**
  Adding stuff needed for io (friend declarations, etc.)
- class **LCRunHeaderIOImpl**
  Adding stuff needed for io (friend declarations, etc.)
- class **MCParticleIOImpl**
  Adding stuff needed for io (friend declarations, etc.)

- class **ParticleIDIOImpl**
  Adding stuff needed for io (friend declarations, etc.)
- class **RawCalorimeterHitIOImpl**
  Adding stuff needed for io (friend declarations, etc.)
- class **ReconstructedParticleIOImpl**
  Adding stuff needed for io (friend declarations, etc.)
- class **SimCalorimeterHitIOImpl**
  Adding stuff needed for io (friend declarations, etc.)
- class **SimTrackerHitIOImpl**
  Adding stuff needed for io (friend declarations, etc.)
- class **TPCHitIOImpl**
  Adding stuff needed for io (friend declarations, etc.)
- class **TrackerDataIOImpl**
  Adding stuff needed for io (friend declarations, etc.)
- class **TrackerHitIOImpl**
  Adding stuff needed for io (friend declarations, etc.)
- class **TrackerHitPlaneIOImpl**
  Adding stuff needed for io (friend declarations, etc.)
- class **TrackerHitZCylinderIOImpl**
  Adding stuff needed for io (friend declarations, etc.)
- class **TrackerPulseIOImpl**
  Adding stuff needed for io (friend declarations, etc.)
- class **TrackerRawDataIOImpl**
  Adding stuff needed for io (friend declarations, etc.)
- class **TrackIOImpl**
  Adding stuff needed for io (friend declarations, etc.)
- class **TrackStateIOImpl**
  Adding stuff needed for io (friend declarations, etc.)
- class **VertexIOImpl**
  Adding stuff needed for io (friend declarations, etc.)

6.5.1 Detailed Description

The **IOIMPL** (p. 27) namespace holds extensions to the default implementations needed for **IO** (p. 27). For internal use only - with the exception of **IOIMPL::LCFactory** (p. 101).

6.6 lcio Namespace Reference

The LCIO namespace combines **EVENT** (p. 21), **IO** (p. 27), **IMPL** (p. 25) and **UTIL** (p. 32) for user convenience.

6.6.1 Detailed Description

The LCIO namespace combines **EVENT** (p. 21), **IO** (p. 27), **IMPL** (p. 25) and **UTIL** (p. 32) for user convenience.

6.7 lcrtrel Namespace Reference

The namespace lcrtrel holds classes and functions for runtime extensions and relations see **LCRTRelations** (p. 142) for more detailed documentation.

Classes

- struct **LCExtension**
  Simple Extension - pointer to an object of type T.
- class **LCOwnedExtension**
  Simple Extension - pointer to an object of type T where the ownership is taken over by the object holding the extension, i.e.
- class **LCExtensionVector**
  Extension vector holding pointers to objects of type T - no ownership of the objects is taken.
- class **LCOwnedExtensionVector**
  Extension vector holding pointers to objects of type T - ownership of the objects is taken, i.e.
- class **LCExtensionList**
  Extension list holding pointers to objects of type T - no ownership of the objects is taken.
- class **LCOwnedExtensionList**
  Extension list holding pointers to objects of type T - ownership of the objects is taken, i.e.
- struct **LC1To1Relation**
  One to one relation between two objects of type From and To.
- struct **LC1ToNRelation**
  One to many relation between one object of type From to many objects of type To.
- struct **LCNToNRelation**
  Many to many relation between objects of type From to objects of type To.
- struct **LCIntExtension**
  Special Extension that allows to write int extensions directly (not through a pointer !).
- struct **LCFloatExtension**
- class **LCRTRelations**
  Base class that provides run time (user) extensions and relation between objects.

Functions

- template<class R >
  void **set_relation** (typename R::from::obj_ptr f, typename R::to::obj_ptr t)
  Set the 1-to-1 relation between two objects - prexisting inconsistent relations involving the two objects are deleted to enforce a consistent set of from-to relations.
- template<class R >
  void **unset_relation** (typename R::from::obj_ptr f)
  Unset the 1-to-1 relation from f.
- template<class R >
  void **add_relation** (typename R::from::obj_ptr f, typename R::to::obj_ptr t)
  Add a link from f to t to an N-to-N relation ship.
- template<class R >
  void **remove_relation** (typename R::from::obj_ptr f, typename R::to::obj_ptr t)
  Remove the link from from f to t from the N-to-N relation ship.
- template<class R >
  void **remove_relations** (typename R::from::obj_ptr f)
  Removes all relations from the given object.
- template<class R >
  void **merge_relations** (typename R::from::obj_ptr f1, typename R::from::obj_ptr f2)
  Merge the relations from f2 to f1 - after this call f1 will hold all the relations and f2 will be empty.

6.7.1 Detailed Description

The namespace lcrtrel holds classes and functions for runtime extensions and relations see **LCRTRelations** (p. 142) for more detailed documentation.

6.7.2 Function Documentation

6.7.2.1 template<class R > void lcrtrel::set_relation (typename R::from::obj_ptr *f*, typename R::to::obj_ptr *t*)

Set the 1-to-1 relation between two objects - prexisting inconsistent relations involving the two objects are deleted to enforce a consistent set of from-to relations.

6.8 SIO Namespace Reference

The **SIO** (p. 30) namespace holds the persistency implementation using **SIO** (p. 30).

Classes

- struct **RunEvent**
  Helper struct that stores run and event positions in the file.
- class **LCIORandomAccess**
  *Implementation class for **LCIORandomAccess** (p. 118) records.*
- class **LCIORandomAccessMgr**
  Manager class for LCIO direct access.
- class **SIORecords**
  Manager class that holds instances of all known LCIO-SIO records.
- class **LCSIO**
  *Collection of static helper functions for reading and writing data with **SIO** (p. 30).*
- class **RunEventMap**
  Map that holds positions of Run and Event records.
- class **SIOCalHitHandler**
  *Implementation of **SIOObjectHandler** (p. 229) to handle **IO** (p. 27) of CalorimeterHits.*
- class **SIOClusterHandler**
  *Implementation of **SIOObjectHandler** (p. 229) to handle **IO** (p. 27) of Clusters.*
- class **SIOCollectionHandler**
  *Handler for LCCollection/LCCollectionIOVec objects for **SIO** (p. 30).*
- class **SIOEventHandler**
  Handler for LCEvent/LCEventIOImpl objects.
- class **SIOFloatVecHandler**
  *Implementation of **SIOObjectHandler** (p. 229) to handle **IO** (p. 27) of FloatVecs.*
- class **SIOHandlerMgr**
  Singleton that holds instances of all types of SIOObjectHandlers.
- class **SIOIndexHandler**
  Handler for LCIOrandomAccess and LCIOIndex objects/blocks.
- class **SIOIntVecHandler**
  *Implementation of **SIOObjectHandler** (p. 229) to handle **IO** (p. 27) of IntVecs.*
- class **SIOLCGenericObjectHandler**
  *Implementation of **SIOObjectHandler** (p. 229) to handle **IO** (p. 27) of LCGenericObjects.*
- class **SIOLCParameters**
  ***IO** (p. 27) of LCParameters.*
- class **SIOLCRelationHandler**
  *Implementation of **SIOObjectHandler** (p. 229) to handle **IO** (p. 27) of CalorimeterHits.*
- class **SIOObjectHandler**
  Interface for all lcio object SIO-handlers, has to be implemented for all event entities (hits, tracks, clusters,...).
- class **SIOParticleHandler**

Implementation of ***SIOObjectHandler*** *(p. 229) to handle* ***IO*** *(p. 27) of MCParticles.*
- class **SIORandomAccessHandler**

  Handler for LCIOrandomAccess and LCIOIndex objects/blocks.
- class **SIORawCalHitHandler**

  Implementation of ***SIOObjectHandler*** *(p. 229) to handle* ***IO*** *(p. 27) of RawCalorimeterHits.*
- class **SIOReader**

  Concrete implementation of LCWriter using ***SIO*** *(p. 30).*
- class **SIOReconstructedParticleHandler**

  Implementation of ***SIOObjectHandler*** *(p. 229) to handle* ***IO*** *(p. 27) of ReconstructedParticles.*
- class **SIORunHeaderHandler**

  Handler for LCRunHeader/LCRunHeaderImpl objects.
- class **SIOSimCalHitHandler**

  Implementation of ***SIOObjectHandler*** *(p. 229) to handle* ***IO*** *(p. 27) of SimCalorimeterHits.*
- class **SIOSimTrackHitHandler**

  Implementation of ***SIOObjectHandler*** *(p. 229) to handle* ***IO*** *(p. 27) of SimTrackerHits.*
- class **SIOStrVecHandler**

  Implementation of ***SIOObjectHandler*** *(p. 229) to handle* ***IO*** *(p. 27) of StrVecs.*
- class **SIOTPCHitHandler**

  Implementation of ***SIOObjectHandler*** *(p. 229) to handle* ***IO*** *(p. 27) of TPCHits.*
- class **SIOTrackerDataHandler**

  Implementation of ***SIOObjectHandler*** *(p. 229) to handle* ***IO*** *(p. 27) of TrackerDatas.*
- class **SIOTrackerHitHandler**

  Implementation of ***SIOObjectHandler*** *(p. 229) to handle* ***IO*** *(p. 27) of SimTrackerHits.*
- class **SIOTrackerHitPlaneHandler**

  Implementation of ***SIOObjectHandler*** *(p. 229) to handle* ***IO*** *(p. 27) of TrackerHitPlane.*
- class **SIOTrackerHitZCylinderHandler**

  Implementation of ***SIOObjectHandler*** *(p. 229) to handle* ***IO*** *(p. 27) of TrackerHitZCylinder.*
- class **SIOTrackerPulseHandler**

  Implementation of ***SIOObjectHandler*** *(p. 229) to handle* ***IO*** *(p. 27) of TrackerPulses.*
- class **SIOTrackerRawDataHandler**

  Implementation of ***SIOObjectHandler*** *(p. 229) to handle* ***IO*** *(p. 27) of TrackerRawDatas.*
- class **SIOTrackHandler**

  Implementation of ***SIOObjectHandler*** *(p. 229) to handle* ***IO*** *(p. 27) of Tracks.*
- class **SIOVertexHandler**

  Implementation of ***SIOObjectHandler*** *(p. 229) to handle* ***IO*** *(p. 27) of Vertexes.*
- class **SIOWriter**

  Concrete implementation of LCWriter using ***SIO*** *(p. 30).*

Typedefs

- typedef std::map< std::string,
  SIOObjectHandler ∗ > **SIOHandlerMap**

Functions

- std::ostream & **operator**<< (std::ostream &os, const **RunEvent** &re)
- bool **operator**< (const **RunEvent** &r0, const **RunEvent** &other)
- std::ostream & **operator**<< (std::ostream &os, const **LCIORandomAccess** &ra)
- bool **operator**< (const **LCIORandomAccess** ra0, const **LCIORandomAccess** &other)
- std::ostream & **operator**<< (std::ostream &os, const **LCIORandomAccessMgr** &ra)
- std::ostream & **operator**<< (std::ostream &os, const **RunEventMap** &rm)

6.8.1 Detailed Description

The **SIO** (p. 30) namespace holds the persistency implementation using **SIO** (p. 30). Users should not use any of the classes defined here explicitly but through their base interfaces defined in **IO** (p. 27).

6.9 UTIL Namespace Reference

The namespace **UTIL** (p. 32) holds convenient and support classes and methods for the LCIO objects.

Classes

- class **LCTokenizer**

  Helper class for string tokenization.
- class **BitFieldValue**

  Helper class for ***BitField64*** *(p. 43) that corresponds to one field value.*
- class **BitField64**

  A bit field of 64bits that allows convenient declaration and manipulation of sub fields of various widths.
- class **BitSet32**

  Convenient helper class for setting single bits in a 32bit-field, such as collection flags or hit types - extends std-::bitset<32>.
- class **CellIDDecoder**

  Convenient class for decoding cellIDs from collection parameter LCIO::CellIDEncoding.
- class **CellIDEncoder**

  Specialization for SimTrackerHits that have only one cellID.
- class **CollectionParameterMap**

  Helper class for setting and retrieving an std::map<string,int> as collection parameters.
- struct **ILDCellID0**

  Helper class to define constants for the canonical encoding of the CellID0 in tracking sub detectors in ILD (or ILD like detectors).
- class **ILDCellIDEncoder**

  Special wrapper to the CellIDEncoder<T> that enforces cellID0 to use the encoder string defined in ***ILDCellID0-::encoder_string*** *(p. 73).*
- struct **ILDDetID**

  Define integer constants for identifying ILD sub detectors.
- struct **ILDTrkHitTypeBit**

  Define integer constants for bits of the the lcio tracker hit type used in the context of ILD reconstruction code.
- struct **ILDTrkHitQualityBit**

  Define integer constants for bits of the the lcio tracker hit quality used in the context of ILD reconstruction code.
- class **IndexMap**

  utility class to manage indexes according to Collection Parameters
- class **LCFixedObject**

  Template class for fixed size LCGenericObject subclasses.
- class **LCFourVector**

  Four vector used in LCIO.
- class **LCIterator**

  Simple convenient iterator class for LCCollections that saves some boiler plate code.
- class **LCObjectHandle**

  Template handle class for LCObjects.
- class **LCRelationNavigator**

  The ***LCRelationNavigator*** *(p. 139) makes repeated lookup of relations more conveneient and efficient.*

- class **LCSplitWriter**
  LCWriter wrapper that automatically splits files if a given number of bytes is exceeded.
- class **LCStdHepRdr**
  Basic utility for reading a binary stdhep file and filling a LCCollectionVec with MCParticles containing the stdhep file information.
- class **LCStdHepRdrNew**
  Basic utility for reading a binary stdhep file and filling a LCCollectionVec with MCParticles containing the stdhep file information.
- class **LCTime**
  Helper class that allows to convert time stamps as defined in LCEvent::getTimeStamp() (ns since 1.1.1970 00:00:00 UTC) to ordinary date and time and vice versa.
- class **LCTOOLS**
  Provides some basic functions for printing event data.
- class **LCTypedVector**
  Templated helper class that is an std::vector< T > of the elements in the LCCollection.*
- class **LCWarning**
  Utility class to show warnings in LCIO.
- class **lcio_short**
- class **LCIO_LONG**
- class **PIDHandler**
  Convenient class for setting and retrieving particle id information attached to ReconstructedParticles.
- class **UnknownAlgorithm**
  Exception for unknown algorithms.

Enumerations

- enum **ObjectType** { **Cluster** = 1, **ReconstructedParticle** }

Functions

- std::ostream & **operator<<** (std::ostream &os, const **BitField64** &b)
  *Operator for dumping **BitField64** (p. 43) to streams.*
- **BitSet32 make_bitset32** (int bit0)
  *Convenient helper that creates a **BitSet32** (p. 47) with bit0 set.*
- **BitSet32 make_bitset32** (int bit0, int bit1)
  *Convenient helper that creates a **BitSet32** (p. 47) with bit0 and bit1 set.*
- **BitSet32 make_bitset32** (int bit0, int bit1, int bit2)
  *Convenient helper that creates a **BitSet32** (p. 47) with bit0 - bit2 set.*
- **BitSet32 make_bitset32** (int bit0, int bit1, int bit2, int bit3)
  *Convenient helper that creates a **BitSet32** (p. 47) with bit0 - bit3 set.*
- **BitSet32 make_bitset32** (int bit0, int bit1, int bit2, int bit3, int bit4)
  *Convenient helper that creates a **BitSet32** (p. 47) with bit0 - bit4 set.*
- int **set_bit** (int flag, int bit)
  Convenient helper to set a bit in a given int - for example:
  trkHit->setType(set_bit(trkHit->getType() , UTIL::ILDTrkHitTypeBit::ONE_DIMENSIONAL)) ;.
- int **unset_bit** (int flag, int bit)
  Convenient helper to unset a bit in a given int - for example:
  trkHit->setType(set_bit(trkHit->getType() , UTIL::ILDTrkHitTypeBit::ONE_DIMENSIONAL)) ;.
- template< class T >
  int **CellIDEncoder_cellID1Bit** ()
  Helper function that returns the bit for cellid1 through template specialization or -1 if no cellid1 exists.

- template<>
  int **CellIDEncoder_cellID1Bit**< **EVENT::SimCalorimeterHit** > ()
  specialization that returns the proper bit for the second cellid
- template<>
  int **CellIDEncoder_cellID1Bit**< **EVENT::RawCalorimeterHit** > ()
  specialization that returns the proper bit for the second cellid
- template<>
  int **CellIDEncoder_cellID1Bit**< **EVENT::CalorimeterHit** > ()
  specialization that returns the proper bit for the second cellid
- template<>
  int **CellIDEncoder_cellID1Bit**< **EVENT::TrackerData** > ()
  specialization that returns the proper bit for the second cellid
- template<>
  int **CellIDEncoder_cellID1Bit**< **EVENT::TrackerPulse** > ()
  specialization that returns the proper bit for the second cellid
- template<>
  int **CellIDEncoder_cellID1Bit**< **EVENT::TrackerRawData** > ()
  specialization that returns the proper bit for the second cellid
- template<>
  int **CellIDEncoder_cellID1Bit**< **EVENT::TrackerHit** > ()
  specialization that returns the proper bit for the second cellid
- template<>
  int **CellIDEncoder_cellID1Bit**< **EVENT::TrackerHitPlane** > ()
  specialization that returns the proper bit for the second cellid
- template<>
  int **CellIDEncoder_cellID1Bit**< **EVENT::TrackerHitZCylinder** > ()
  specialization that returns the proper bit for the second cellid
- template<>
  int **CellIDEncoder_cellID1Bit**< **EVENT::SimTrackerHit** > ()
  specialization that returns the proper bit for the second cellid
- template< class T >
  void **CellIDEncoder_setCellID** (T *hit, int low, int high)
  Helper function that sets cellid1 and cellid2.
- template< class T >
  const char * **lctypename** ()
  Template that returns the LCIO type name as used in the LCCollctions (and files), e.g.
- template<>
  const char * **lctypename**< **EVENT::MCParticle** > ()
- template<>
  const char * **lctypename**< **EVENT::SimCalorimeterHit** > ()
- template<>
  const char * **lctypename**< **EVENT::CalorimeterHit** > ()
- template<>
  const char * **lctypename**< **EVENT::RawCalorimeterHit** > ()
- template<>
  const char * **lctypename**< **EVENT::SimTrackerHit** > ()
- template<>
  const char * **lctypename**< **EVENT::TrackerRawData** > ()
- template<>
  const char * **lctypename**< **EVENT::TrackerData** > ()
- template<>
  const char * **lctypename**< **EVENT::TrackerPulse** > ()
- template<>
  const char * **lctypename**< **EVENT::TrackerHit** > ()

- template<>
  const char ∗ **lctypename**< **EVENT::TrackerHitPlane** > ()
- template<>
  const char ∗ **lctypename**< **EVENT::TrackerHitZCylinder** > ()
- template<>
  const char ∗ **lctypename**< **EVENT::LCStrVec** > ()
- template<>
  const char ∗ **lctypename**< **EVENT::LCFloatVec** > ()
- template<>
  const char ∗ **lctypename**< **EVENT::LCIntVec** > ()
- template<>
  const char ∗ **lctypename**< **EVENT::Track** > ()
- template<>
  const char ∗ **lctypename**< **EVENT::Cluster** > ()
- template<>
  const char ∗ **lctypename**< **EVENT::ReconstructedParticle** > ()
- template<>
  const char ∗ **lctypename**< **EVENT::LCRelation** > ()
- template<>
  const char ∗ **lctypename**< **EVENT::LCGenericObject** > ()
- template<>
  const char ∗ **lctypename**< **EVENT::Vertex** > ()
- template<class T >
  const char ∗ **lctypename** (const T ∗t)

  Template that returns the LCIO type name as used in the LCCollctions (and files) based on the object's type, e.g.
- template<>
  const char ∗ **lctypename**< **EVENT::LCObject** > (const **EVENT::LCObject** ∗o)

  Specialization for polymorphic LCObject pointers - slow.
- template<class T >
  lcio_short< T > **lcshort** (const T ∗t, const **EVENT::LCCollection** ∗b=NULL)
- const std::string & **header** (const **EVENT::Vertex** ∗)

  operator for detailed output of a vertex object (e.g.
- const std::string & **tail** (const **EVENT::Vertex** ∗)
- std::ostream & **operator**<< (std::ostream &out, const **UTIL::lcio_short**< **EVENT::Vertex** > &sV)
- std::ostream & **operator**<< (std::ostream &out, const **EVENT::Vertex** &v)
- const std::string & **header** (const **EVENT::MCParticle** ∗)
- const std::string & **tail** (const **EVENT::MCParticle** ∗)
- std::ostream & **operator**<< (std::ostream &out, const **UTIL::lcio_short**< **EVENT::MCParticle** > &sV)
- std::ostream & **operator**<< (std::ostream &out, const **EVENT::MCParticle** &)
- const std::string & **header** (const **EVENT::TrackerHit** ∗)
- const std::string & **tail** (const **EVENT::TrackerHit** ∗)
- std::ostream & **operator**<< (std::ostream &out, const **UTIL::lcio_short**< **EVENT::TrackerHit** > &sV)
- std::ostream & **operator**<< (std::ostream &out, const **EVENT::TrackerHit** &)
- const std::string & **header** (const **EVENT::TrackerHitPlane** ∗)
- const std::string & **tail** (const **EVENT::TrackerHitPlane** ∗)
- std::ostream & **operator**<< (std::ostream &out, const **UTIL::lcio_short**< **EVENT::TrackerHitPlane** > &sV)
- std::ostream & **operator**<< (std::ostream &out, const **EVENT::TrackerHitPlane** &)
- const std::string & **header** (const **EVENT::TrackerHitZCylinder** ∗)
- const std::string & **tail** (const **EVENT::TrackerHitZCylinder** ∗)
- std::ostream & **operator**<< (std::ostream &out, const **UTIL::lcio_short**< **EVENT::TrackerHitZCylinder** > &sV)
- std::ostream & **operator**<< (std::ostream &out, const **EVENT::TrackerHitZCylinder** &)
- const std::string & **header** (const **EVENT::SimTrackerHit** ∗)
- const std::string & **tail** (const **EVENT::SimTrackerHit** ∗)
- std::ostream & **operator**<< (std::ostream &out, const **UTIL::lcio_short**< **EVENT::SimTrackerHit** > &sV)

- std::ostream & **operator**<< (std::ostream &out, const **EVENT::SimTrackerHit** &)
- const std::string & **header** (const **EVENT::CalorimeterHit** ∗)
- const std::string & **tail** (const **EVENT::CalorimeterHit** ∗)
- std::ostream & **operator**<< (std::ostream &out, const **UTIL::lcio_short**< **EVENT::CalorimeterHit** > &sV)
- std::ostream & **operator**<< (std::ostream &out, const **EVENT::CalorimeterHit** &)
- const std::string & **header** (const **EVENT::SimCalorimeterHit** ∗)
- const std::string & **tail** (const **EVENT::SimCalorimeterHit** ∗)
- std::ostream & **operator**<< (std::ostream &out, const **UTIL::lcio_short**< **EVENT::SimCalorimeterHit** > &sV)
- std::ostream & **operator**<< (std::ostream &out, const **EVENT::SimCalorimeterHit** &)
- const std::string & **header** (const **EVENT::ReconstructedParticle** ∗)
- const std::string & **tail** (const **EVENT::ReconstructedParticle** ∗)
- std::ostream & **operator**<< (std::ostream &out, const **UTIL::lcio_short**< **EVENT::ReconstructedParticle** > &)
- std::ostream & **operator**<< (std::ostream &out, const **EVENT::ReconstructedParticle** &)
- const std::string & **header** (const **EVENT::Track** ∗)
- const std::string & **tail** (const **EVENT::Track** ∗)
- std::ostream & **operator**<< (std::ostream &out, const **UTIL::lcio_short**< **EVENT::Track** > &)
- std::ostream & **operator**<< (std::ostream &out, const **EVENT::Track** &)
- const std::string & **header** (const **EVENT::TrackState** ∗)
- const std::string & **tail** (const **EVENT::TrackState** ∗)
- std::ostream & **operator**<< (std::ostream &out, const **UTIL::lcio_short**< **EVENT::TrackState** > &)
- std::ostream & **operator**<< (std::ostream &out, const **EVENT::TrackState** &)
- const std::string & **header** (const **EVENT::Cluster** ∗)
- const std::string & **tail** (const **EVENT::Cluster** ∗)
- std::ostream & **operator**<< (std::ostream &out, const **UTIL::lcio_short**< **EVENT::Cluster** > &)
- std::ostream & **operator**<< (std::ostream &out, const **EVENT::Cluster** &)
- const std::string & **header** (const **EVENT::LCRelation** ∗)
- const std::string & **tail** (const **EVENT::LCRelation** ∗)
- std::ostream & **operator**<< (std::ostream &out, const **UTIL::lcio_short**< **EVENT::LCRelation** > &)
- std::ostream & **operator**<< (std::ostream &out, const **EVENT::LCRelation** &)
- const std::string & **header** (const **EVENT::LCFloatVec** ∗)
- const std::string & **tail** (const **EVENT::LCFloatVec** ∗)
- std::ostream & **operator**<< (std::ostream &out, const **UTIL::lcio_short**< **EVENT::LCFloatVec** > &)
- std::ostream & **operator**<< (std::ostream &out, const **EVENT::LCFloatVec** &)
- template<class T >
  const std::string & **header** ()
- template<class T >
  const std::string & **tail** ()
- template<class T >
  LCIO_LONG< T > **lcio_long** (const T &o, const **EVENT::LCCollection** ∗c)
- std::ostream & **operator**<< (std::ostream &out, const **LCIO_LONG**< **EVENT::CalorimeterHit** > l)
- std::ostream & **operator**<< (std::ostream &out, const **LCIO_LONG**< **EVENT::Cluster** > l)
- std::ostream & **operator**<< (std::ostream &out, const **LCIO_LONG**< **EVENT::Track** > l)
- std::ostream & **operator**<< (std::ostream &out, const **LCIO_LONG**< **EVENT::TrackState** > l)
- std::ostream & **operator**<< (std::ostream &out, const **LCIO_LONG**< **EVENT::ReconstructedParticle** > ll)
- std::ostream & **operator**<< (std::ostream &out, const **LCIO_LONG**< **EVENT::SimCalorimeterHit** > ll)
- std::ostream & **operator**<< (std::ostream &out, const **LCIO_LONG**< **EVENT::TrackerHit** > ll)
- std::ostream & **operator**<< (std::ostream &out, const **LCIO_LONG**< **EVENT::TrackerHitPlane** > ll)
- std::ostream & **operator**<< (std::ostream &out, const **LCIO_LONG**< **EVENT::TrackerHitZCylinder** > ll)
- std::ostream & **operator**<< (std::ostream &out, const **LCIO_LONG**< **EVENT::SimTrackerHit** > ll)
- std::ostream & **operator**<< (std::ostream &out, const **LCIO_LONG**< **EVENT::MCParticle** > ll)
- std::ostream & **operator**<< (std::ostream &out, const **LCIO_LONG**< **EVENT::Vertex** > ll)
- std::ostream & **operator**<< (std::ostream &out, const **LCIO_LONG**< **EVENT::LCRelation** > ll)
- std::ostream & **operator**<< (std::ostream &out, const **LCIO_LONG**< **EVENT::LCFloatVec** > ll)

- const std::string & **header** (const **EVENT::LCCollection** ∗)
- const std::string & **tail** (const **EVENT::LCCollection** ∗)
- std::ostream & **operator**<< (std::ostream &out, const **UTIL::lcio_short**< **EVENT::LCCollection** > &)
- std::ostream & **operator**<< (std::ostream &out, const **EVENT::LCCollection** &)
- const std::string & **header** (const **EVENT::LCEvent** ∗)
- const std::string & **tail** (const **EVENT::LCEvent** ∗)
- std::ostream & **operator**<< (std::ostream &out, const **UTIL::lcio_short**< **EVENT::LCEvent** > &)
- std::ostream & **operator**<< (std::ostream &out, const **EVENT::LCEvent** &)
- const std::string & **header** (const **EVENT::LCFlag** ∗)
- const std::string & **tail** (const **EVENT::LCFlag** ∗)
- std::ostream & **operator**<< (std::ostream &out, const **UTIL::lcio_short**< **EVENT::LCFlag** > &)
- std::ostream & **operator**<< (std::ostream &out, const **EVENT::LCFlag** &)
- const std::string & **header** (const **EVENT::LCGenericObject** ∗, const **EVENT::LCCollection** ∗v=NULL)
- const std::string & **tail** (const **EVENT::LCGenericObject** ∗)
- std::ostream & **operator**<< (std::ostream &out, const **UTIL::lcio_short**< **EVENT::LCGenericObject** > &)
- std::ostream & **operator**<< (std::ostream &out, const **EVENT::LCGenericObject** &)
- const std::string & **header** (const **EVENT::LCIntVec** ∗)
- const std::string & **tail** (const **EVENT::LCIntVec** ∗)
- std::ostream & **operator**<< (std::ostream &out, const **UTIL::lcio_short**< **EVENT::LCIntVec** > &)
- std::ostream & **operator**<< (std::ostream &out, const **EVENT::LCIntVec** &)
- const std::string & **header** (const **EVENT::LCObject** ∗)
- const std::string & **tail** (const **EVENT::LCObject** ∗)
- std::ostream & **operator**<< (std::ostream &out, const **UTIL::lcio_short**< **EVENT::LCObject** > &)
- std::ostream & **operator**<< (std::ostream &out, const **EVENT::LCObject** &)
- const std::string & **header** (const **EVENT::LCParameters** ∗)
- const std::string & **tail** (const **EVENT::LCParameters** ∗)
- std::ostream & **operator**<< (std::ostream &out, const **UTIL::lcio_short**< **EVENT::LCParameters** > &)
- std::ostream & **operator**<< (std::ostream &out, const **EVENT::LCParameters** &)
- const std::string & **header** (const **EVENT::LCRunHeader** ∗)
- const std::string & **tail** (const **EVENT::LCRunHeader** ∗)
- std::ostream & **operator**<< (std::ostream &out, const **UTIL::lcio_short**< **EVENT::LCRunHeader** > &)
- std::ostream & **operator**<< (std::ostream &out, const **EVENT::LCRunHeader** &)
- const std::string & **header** (const **EVENT::ParticleID** ∗)
- const std::string & **tail** (const **EVENT::ParticleID** ∗)
- std::ostream & **operator**<< (std::ostream &out, const **UTIL::lcio_short**< **EVENT::ParticleID** > &)
- std::ostream & **operator**<< (std::ostream &out, const **EVENT::ParticleID** &)
- const std::string & **header** (const **EVENT::RawCalorimeterHit** ∗)
- const std::string & **tail** (const **EVENT::RawCalorimeterHit** ∗)
- std::ostream & **operator**<< (std::ostream &out, const **UTIL::lcio_short**< **EVENT::RawCalorimeterHit** > &)
- std::ostream & **operator**<< (std::ostream &out, const **EVENT::RawCalorimeterHit** &)
- const std::string & **header** (const **EVENT::TrackerData** ∗)
- const std::string & **tail** (const **EVENT::TrackerData** ∗)
- std::ostream & **operator**<< (std::ostream &out, const **UTIL::lcio_short**< **EVENT::TrackerData** > &)
- std::ostream & **operator**<< (std::ostream &out, const **EVENT::TrackerData** &)
- const std::string & **header** (const **EVENT::TrackerPulse** ∗)
- const std::string & **tail** (const **EVENT::TrackerPulse** ∗)
- std::ostream & **operator**<< (std::ostream &out, const **UTIL::lcio_short**< **EVENT::TrackerPulse** > &)
- std::ostream & **operator**<< (std::ostream &out, const **EVENT::TrackerPulse** &)
- const std::string & **header** (const **EVENT::TrackerRawData** ∗)
- const std::string & **tail** (const **EVENT::TrackerRawData** ∗)
- std::ostream & **operator**<< (std::ostream &out, const **UTIL::lcio_short**< **EVENT::TrackerRawData** > &)
- std::ostream & **operator**<< (std::ostream &out, const **EVENT::TrackerRawData** &)
- const std::string & **header** (const **EVENT::LCIO** ∗)
- const std::string & **tail** (const **EVENT::LCIO** ∗)

- std::ostream & **operator**<< (std::ostream &out, const **UTIL::lcio_short**< **EVENT::LCIO** > &)
- std::ostream & **operator**<< (std::ostream &out, const **EVENT::LCIO** &)
- template<class T >
  std::string **toString** (const T ∗obj)
- std::ostream & **operator**<< (std::ostream &out, const **LCIO_LONG**< **EVENT::LCCollection** > l)
- std::ostream & **operator**<< (std::ostream &out, const **LCIO_LONG**< **EVENT::LCEvent** > l)
- std::ostream & **operator**<< (std::ostream &out, const **LCIO_LONG**< **EVENT::LCFlag** > l)
- std::ostream & **operator**<< (std::ostream &out, const **LCIO_LONG**< **EVENT::LCGenericObject** > l)
- std::ostream & **operator**<< (std::ostream &out, const **LCIO_LONG**< **EVENT::LCIntVec** > l)
- std::ostream & **operator**<< (std::ostream &out, const **LCIO_LONG**< **EVENT::LCObject** > l)
- std::ostream & **operator**<< (std::ostream &out, const **LCIO_LONG**< **EVENT::LCParameters** > l)
- std::ostream & **operator**<< (std::ostream &out, const **LCIO_LONG**< **EVENT::LCRunHeader** > l)
- std::ostream & **operator**<< (std::ostream &out, const **LCIO_LONG**< **EVENT::LCIO** > l)
- std::ostream & **operator**<< (std::ostream &out, const **LCIO_LONG**< **EVENT::ParticleID** > l)
- std::ostream & **operator**<< (std::ostream &out, const **LCIO_LONG**< **EVENT::RawCalorimeterHit** > l)
- std::ostream & **operator**<< (std::ostream &out, const **LCIO_LONG**< **EVENT::TrackerData** > l)
- std::ostream & **operator**<< (std::ostream &out, const **LCIO_LONG**< **EVENT::TrackerPulse** > l)
- std::ostream & **operator**<< (std::ostream &out, const **LCIO_LONG**< **EVENT::TrackerRawData** > l)

Variables

- static int **MAX_HITS** = 1000

6.9.1 Detailed Description

The namespace **UTIL** (p. 32) holds convenient and support classes and methods for the LCIO objects.

6.9.2 Function Documentation

6.9.2.1 template<> int UTIL::CellIDEncoder_cellID1Bit< EVENT::SimCalorimeterHit >( )

specialization that returns the proper bit for the second cellid

Helper function that returns the bit for cellid1 through template specialization or -1 if no cellid1 exists.

specialization that returns the proper bit for the second cellid

6.9.2.2 const std::string & UTIL::header (const EVENT::Vertex ∗)

operator for detailed output of a vertex object (e.g.

cout << vertexObj << endl;) for printing the header of a vertex object (e.g. cout << header(vertexObj) << endl;) for printing the tail of a vertex object (e.g. cout << tail(vertexObj) << endl;)

Referenced by UTIL::LCTOOLS::printCalorimeterHits(), UTIL::LCTOOLS::printClusters(), UTIL::LCTOOLS::print-LCFloatVecs(), UTIL::LCTOOLS::printLCGenericObjects(), UTIL::LCTOOLS::printLCIntVecs(), UTIL::LCTOOLS-::printRawCalorimeterHits(), UTIL::LCTOOLS::printReconstructedParticles(), UTIL::LCTOOLS::printRelation(), U-TIL::LCTOOLS::printSimCalorimeterHits(), UTIL::LCTOOLS::printSimTrackerHits(), UTIL::LCTOOLS::printTracker-Data(), UTIL::LCTOOLS::printTrackerHitPlane(), UTIL::LCTOOLS::printTrackerHits(), UTIL::LCTOOLS::print-TrackerHitZCylinder(), UTIL::LCTOOLS::printTrackerPulse(), UTIL::LCTOOLS::printTrackerRawData(), UTIL::L-CTOOLS::printTracks(), and UTIL::LCTOOLS::printVertices().

6.9.2.3 template< class T > const char ∗ UTIL::lctypename ( )

Template that returns the LCIO type name as used in the LCCollctions (and files), e.g.

Specialization for implemention classes.

std::cout << **lctypename<MCParticle>()** (p. 39) << std::endl ;

This can be used in user templates to create a new collection of the template type.

6.9.2.4 template< class T > const char∗ UTIL::lctypename (const T ∗ *t*)

Template that returns the LCIO type name as used in the LCCollctions (and files) based on the object's type, e.g.

MCParticle∗ mcp ;

std::cout << lctypename(mcp) << std::endl ;

6.9.2.5 template<> const char∗ UTIL::lctypename< EVENT::LCObject > (const EVENT::LCObject ∗ *o*)

Specialization for polymorphic LCObject pointers - slow.

E.g.:

LCObject∗ obj = new McParticle ;

std::cout << lctypename(obj) << std::endl ;

Chapter 7

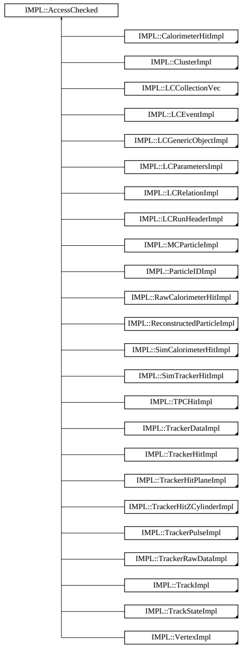
Class Documentation

7.1 IMPL::AccessChecked Class Reference

Controls access to objects.

`#include <IMPL/AccessChecked.h>`

Inheritance diagram for IMPL::AccessChecked:



Public Member Functions

- virtual int **simpleUID** () const

Protected Member Functions

- virtual void **setReadOnly** (bool readOnly)
- void **checkAccess** () throw (EVENT::ReadOnlyException)
- void **checkAccess** (const char ∗what) throw (EVENT::ReadOnlyException)

Protected Attributes

- bool **_readOnly**
- int **_id**

Friends

- class **LCEventImpl**
- class **LCCollectionVec**

7.1.1 Detailed Description

Controls access to objects.

Author

gaede

Version

Sep 4, 2003

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

- AccessChecked.h
- AccessChecked.cc

7.2 lcrtrel_helper::BiDirectional< From, To > Struct Template Reference

Helper class for biderectional relations provides the to and from type.

`#include <LCRTRelations.h>`

Public Types

- typedef From **from**
- typedef To **to**

7.2.1 Detailed Description

template<class From, class To>struct lcrtrel_helper::BiDirectional< From, To >

Helper class for biderectional relations provides the to and from type.

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

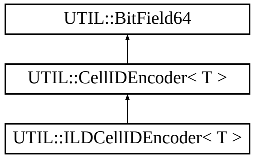
- LCRTRelations.h

7.3 UTIL::BitField64 Class Reference

A bit field of 64bits that allows convenient declaration and manipulation of sub fields of various widths.

```
#include <UTIL/BitField64.h>
```

Inheritance diagram for UTIL::BitField64:



Public Types

- typedef std::map< std::string,
  unsigned int > **IndexMap**

Public Member Functions

- **BitField64** (const std::string &initString)

  The c'tor takes an initialization string of the form:
  <fieldDesc>[,<fieldDesc>...]
  fieldDesc = name:[start]:[-]length
  where:
  name: The name of the field
  start: The start bit of the field.
- **lcio::long64 getValue** () const

  Returns the current 64bit value.
- void **setValue** (**lcio::long64** value)

  Set a new 64bit value.
- void **reset** ()

  Reset - same as setValue(0) - useful if the same encoder is used for many objects.
- **BitFieldValue** & **operator[ ]** (size_t theIndex)

  Acces to field through index.
- const **BitFieldValue** & **operator[ ]** (size_t theIndex) const

  Const acces to field through index.
- unsigned **highestBit** () const

  Highest bit used in fields [0-63].
- size_t **size** () const

  Number of values.
- size_t **index** (const std::string &name) const

  ***Index** (p. 77) for field named 'name'.*
- **BitFieldValue** & **operator[ ]** (const std::string &name)

  Access to field through name .
- const **BitFieldValue** & **operator[ ]** (const std::string &name) const

  Const Access to field through name .
- unsigned **lowWord** () const

  The low word, bits 0-31.
- unsigned **highWord** () const

  The high word, bits 32-63.
- std::string **fieldDescription** () const

  Return a valid description string of all fields.
- std::string **valueString** () const

  Return a string with a comma separated list of the current sub field values.

Protected Member Functions

- void **addField** (const std::string &name, unsigned offset, int width)

  Add an additional field to the list.
- void **init** (const std::string &initString)

  Decode the initialization string as described in the constructor.
- **BitField64** ()

  No default c'tor.

Protected Attributes

- std::vector< **BitFieldValue** $*$ > **_fields**
- **lcio::long64 _value**
- IndexMap **_map**
- **lcio::long64 _joined**

Friends

- std::ostream & **operator**<< (std::ostream &os, const **BitField64** &b)

  *Operator for dumping **BitField64** (p. 43) to streams.*

7.3.1 Detailed Description

A bit field of 64bits that allows convenient declaration and manipulation of sub fields of various widths.

Example:

BitField64 (p. 43) b("layer:7,system:-3,barrel:3,theta:32:11,phi:11") ;

b["layer"] = 123 ;

b["system"] = -4 ;

b["barrel"] = 7 ;

b["theta"] = 180 ;

b["phi"] = 270 ;

...

int theta = b["theta"] ;

...

unsigned phiIndex = b.index("phi) ;

int phi = b[phiIndex] ;

7.3.2 Constructor & Destructor Documentation

7.3.2.1 UTIL::BitField64::BitField64 (const std::string & *initString*) [inline]

The c'tor takes an initialization string of the form:

<fieldDesc>[,<fieldDesc>...]

fieldDesc = name:[start]:[-]length

where:

name: The name of the field

start: The start bit of the field.

If omitted assumed to start immediately following previous field, or at the least significant bit if the first field.

length: The number of bits in the field. If preceeded by '-' the field is signed, otherwise unsigned.

Bit numbering is from the least significant bit (bit 0) to the most significant (bit 63).

Example: "layer:7,system:-3,barrel:3,theta:32:11,phi:11"

References init().

7.3.3 Member Function Documentation

7.3.3.1 void UTIL::BitField64::init (const std::string & *initString*) [protected]

Decode the initialization string as described in the constructor.

See Also

BitField64(const std::string& initString) (p. 45)

References addField().

Referenced by BitField64().

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

- BitField64.h
- BitField64.cc

7.4 UTIL::BitFieldValue Class Reference

Helper class for **BitField64** (p. 43) that corresponds to one field value.

`#include <UTIL/BitField64.h>`

Public Member Functions

- **BitFieldValue** (**lcio::long64** &bitfield, const std::string &**name**, unsigned **offset**, int signedWidth)
  The default c'tor.
- **lcio::long64 value** () const
  Returns the current field value.
- **BitFieldValue** & **operator=** (**lcio::long64** in)
  Assignment operator for user convenience.
- **operator lcio::long64** () const

  Conversion operator for lcio::long64 - allows to write:
  lcio::long64 index = myBitFieldValue ;.
- const std::string & **name** () const
  fg: removed because it causes ambiguities with operator lcio::long64().
- unsigned **offset** () const
  The field's offset.
- unsigned **width** () const
  The field's width.
- bool **isSigned** () const
  True if field is interpreted as signed.
- **lcio::ulong64 mask** () const
  The field's mask.

Protected Attributes

- **lcio::long64** & **_b**
- **lcio::ulong64 _mask**
- std::string **_name**
- unsigned **_offset**
- unsigned **_width**
- int **_minVal**
- int **_maxVal**
- bool **_isSigned**

7.4.1 Detailed Description

Helper class for **BitField64** (p. 43) that corresponds to one field value.

7.4.2 Constructor & Destructor Documentation

7.4.2.1 UTIL::BitFieldValue::BitFieldValue (lcio::long64 & *bitfield*, const std::string & *name*, unsigned *offset*, int *signedWidth*)

The default c'tor.

Parameters

bitfield	reference to the 64bit bitfield
offset	offset of field
signedWidth	width of field, negative if field is signed

References offset().

7.4.3 Member Function Documentation

7.4.3.1 const std::string& UTIL::BitFieldValue::name ( ) const [inline]

fg: removed because it causes ambiguities with operator lcio::long64().

Conversion operator for int - allows to write:

int index = myBitFieldValue ;The field's name

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

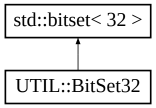
- BitField64.h
- BitField64.cc

7.5 UTIL::BitSet32 Class Reference

Convenient helper class for setting single bits in a 32bit-field, such as collection flags or hit types - extends std::bitset<32>.

```
#include <UTIL/BitSet32.h>
```

Inheritance diagram for UTIL::BitSet32:



Public Member Functions

- **BitSet32** (int val)

  Allow to construct this from an int.
- **BitSet32** & **operator=** (int val)

  Allow assignment from an int.
- **operator int** ()

  Direct conversion to int.

7.5.1 Detailed Description

Convenient helper class for setting single bits in a 32bit-field, such as collection flags or hit types - extends std::bitset<32>.

Use convenient methods make_bitset32(...) to create a bit set with some bits set.

Examples:

```
// set collection flags :
LCCollectionVec* calVec = new LCCollectionVec( LCIO::SIMCALORIMETERHIT )  ;
calVec->setFlag( UTIL::make_bitset32(  LCIO::CHBIT_LONG, LCIO::CHBIT_STEP ) );

// check TrackerHit type:
BitSet32 (p.47) type( hit->getType() ) ;
bool isComposite = type.test( ILDTrkHitTypeBit::COMPOSITE_SPACEPOINT ) ; // test w/ range check
// or shorter:
if( BitSet32 (p.47)( hit->getType() )[ ILDTrkHitTypeBit::COMPOSITE_SPACEPOINT ] ) {...}
```

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

- BitSet32.h

7.6 UTIL::LCTime::CalendarTime Struct Reference

Helper struct that holds the calendar time.

```
#include <UTIL/LCTime.h>
```


Public Attributes

- short **year**
- short **month**
- short **day**
- short **hour**
- short **min**
- short **sec**
- int **ns**

7.6.1 Detailed Description

Helper struct that holds the calendar time.

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

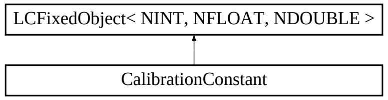
- LCTime.h

7.7 CalibrationConstant Class Reference

Example for a simple calibration class based on the LCFixedObject template.

```
#include </opt/ilcsoft/HEAD-2016-12-02/lcio/HEAD/src/cpp/src/EXAMPLE/Calibration-
Constant.h>
```

Inheritance diagram for CalibrationConstant:



Public Member Functions

- **CalibrationConstant** (int cellID, float offset, float gain)

  Convenient c'tor.
- **CalibrationConstant** (**LCObject** ∗obj)

  'Copy constructor' needed to interpret LCCollection read from file/database.
- virtual ∼**CalibrationConstant** ()

  Important for memory handling.
- int **getCellID** ()
- float **getOffset** ()
- float **getGain** ()
- void **print** (std::ostream &os)
- const std::string **getTypeName** () const
- const std::string **getDataDescription** () const

7.7.1 Detailed Description

Example for a simple calibration class based on the LCFixedObject template.

LCFixedObject uses an instance of LCGenericObjectImpl that holds the data, thus there is no overhead when the data is read from a database or file for copying it to some local structure (Decorator pattern).

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

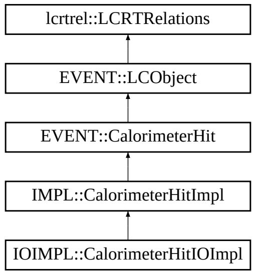
- CalibrationConstant.h

7.8 EVENT::CalorimeterHit Class Reference

The generic calorimeter hit - created from **SimCalorimeterHit** (p. 205) or **RawCalorimeterHit** (p. 188).

```
#include <pre-generated/EVENT/CalorimeterHit.h>
```

Inheritance diagram for EVENT::CalorimeterHit:



Public Types

- typedef **CalorimeterHit lcobject_type**
  *Useful typedef for template programming with **LCIO** (p. 115).*

Public Member Functions

- virtual ∼**CalorimeterHit** ()
  Destructor.
- virtual int **getCellID0** () const =0
  Returns the detector specific (geometrical) cell id.
- virtual int **getCellID1** () const =0
  Returns the second detector specific (geometrical) cell id.
- virtual float **getEnergy** () const =0
  Returns the energy of the hit in [GeV].
- virtual float **getEnergyError** () const =0
  Returns the error of the hit energy in [GeV].
- virtual float **getTime** () const =0
  Returns the time of the hit in [ns].
- virtual const float ∗ **getPosition** () const =0
  Returns the position of the hit in world coordinates.
- virtual int **getType** () const =0

  Type of hit.
- virtual **LCObject** ∗ **getRawHit** () const =0
  *The **RawCalorimeterHit** (p. 188).*

Additional Inherited Members

7.8.1 Detailed Description

The generic calorimeter hit - created from **SimCalorimeterHit** (p. 205) or **RawCalorimeterHit** (p. 188).

The difference to **RawCalorimeterHit** (p. 188) is that amplitude (energy) and time are now float and an optional position (x,y,z) can be stored with the hit. In order to suppress a (32-bit) pointer tag stored with every hit set flag(LCIO::RCHBIT_NO_PTR)==1.

Author

gaede

Version

Id:

CalorimeterHit.aid,v 1.16 2008-06-04 17:37:15 engels Exp

7.8.2 Member Function Documentation

7.8.2.1 virtual int EVENT::CalorimeterHit::getCellID1 ( ) const [pure virtual]

Returns the second detector specific (geometrical) cell id.

Optional, check/set flag(LCIO::RCHBIT_ID1)==1.

Implemented in **IMPL::CalorimeterHitImpl** (p. 53).

Referenced by SIO::SIOCalHitHandler::write().

7.8.2.2 virtual float EVENT::CalorimeterHit::getEnergyError ( ) const [pure virtual]

Returns the error of the hit energy in [GeV].

Optional, check/set flag(LCIO::RCHBIT_ENERGY_ERROR)==1.

Implemented in **IMPL::CalorimeterHitImpl** (p. 53).

Referenced by SIO::SIOCalHitHandler::write().

7.8.2.3 virtual const float∗ EVENT::CalorimeterHit::getPosition ( ) const [pure virtual]

Returns the position of the hit in world coordinates.

NULL if information is not stored - check flag(LCIO::RCHBIT_LONG)==1.

Implemented in **IMPL::CalorimeterHitImpl** (p. 53).

Referenced by SIO::SIOCalHitHandler::write().

7.8.2.4 virtual LCObject∗ EVENT::CalorimeterHit::getRawHit ( ) const [pure virtual]

The **RawCalorimeterHit** (p. 188).

As in the future there might be other types of raw calorimeter hits the return type is **LCObject** (p. 123).

Implemented in **IMPL::CalorimeterHitImpl** (p. 53).

Referenced by SIO::SIOCalHitHandler::write().

7.8.2.5 virtual float EVENT::CalorimeterHit::getTime ( ) const [pure virtual]

Returns the time of the hit in [ns].

Optional, check/set flag(LCIO::RCHBIT_TIME)==1.

Implemented in **IMPL::CalorimeterHitImpl** (p. 54).

Referenced by SIO::SIOCalHitHandler::write().

7.8.2.6 virtual int EVENT::CalorimeterHit::getType ( ) const [pure virtual]

Type of hit.

Mapping of integer types to type names through collection parameters "CalorimeterHitTypeNames" and "-CalorimeterHitTypeValues".

Implemented in **IMPL::CalorimeterHitImpl** (p. 54).

Referenced by SIO::SIOCalHitHandler::write().

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

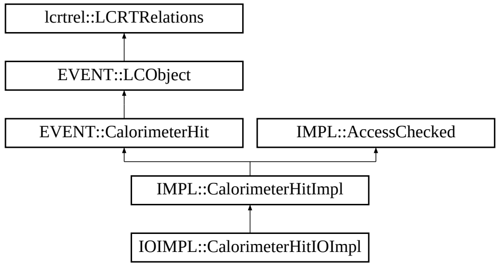
- CalorimeterHit.h

7.9 IMPL::CalorimeterHitImpl Class Reference

Implementation of the real data CalorimeterHit.

`#include <IMPL/CalorimeterHitImpl.h>`

Inheritance diagram for IMPL::CalorimeterHitImpl:



Public Member Functions

- **CalorimeterHitImpl** ()

    Default constructor, initializes values to 0.
- virtual ∼**CalorimeterHitImpl** ()

    Copy constructor.
- virtual int **id** () const

    Returns an object id for internal (debugging) use in LCIO.
- virtual int **getCellID0** () const

    Returns the detector specific (geometrical) cell id.
- virtual int **getCellID1** () const

    Returns the second detector specific (geometrical) cell id.
- virtual float **getEnergy** () const

    Returns the energy of the hit.
- virtual float **getEnergyError** () const

    Returns the error of the hit energy.
- virtual float **getTime** () const

    Returns the time of the hit in [ns].
- virtual const float ∗ **getPosition** () const

    Returns the position of the hit in world coordinates.
- virtual int **getType** () const

    Type of hit.
- virtual **EVENT::LCObject** ∗ **getRawHit** () const

    The RawCalorimeterHit.
- void **setCellID0** (int id0)

    Sets the first cell id;.
- void **setCellID1** (int id1)

    Sets the second cell id; Only store if the flag word (bit RCHBIT_ID1) of the collection is set.
- void **setEnergy** (float en)

    Sets the energy/amplitude.
- void **setEnergyError** (float enerr)

    Sets the error of the hit energy.
- void **setTime** (float t)

    Sets the time.
- void **setPosition** (const float pos[3])

    Sets the position.
- void **setType** (int type)
- void **setRawHit** (**EVENT::LCObject** ∗rawHit)

Protected Attributes

- int **_cellID0**
- int **_cellID1**
- float **_energy**
- float **_energyError**
- float **_time**
- float **_position** [3]
- int **_type**
- **EVENT::LCObject** ∗ **_rawHit**

Additional Inherited Members

7.9.1 Detailed Description

Implementation of the real data CalorimeterHit.

Author

gaede

Version

Id:

CalorimeterHitImpl.h (p. **??**),v 1.16 2008-06-04 17:37:16 engels Exp

7.9.2 Constructor & Destructor Documentation

7.9.2.1 IMPL::CalorimeterHitImpl::∼CalorimeterHitImpl ( ) [virtual]

Copy constructor.

Not yet - needs pointer chasing ... Destructor.

7.9.3 Member Function Documentation

7.9.3.1 int IMPL::CalorimeterHitImpl::getCellID1 ( ) const [virtual]

Returns the second detector specific (geometrical) cell id.

0 if information is not stored - check the flag word (bit RCHBIT_ID1) of the collection. Default is to store only cellid0.

Implements **EVENT::CalorimeterHit** (p. 50).

7.9.3.2 float IMPL::CalorimeterHitImpl::getEnergyError ( ) const [virtual]

Returns the error of the hit energy.

Optional, check/set flag(LCIO::RCHBIT_ENERGY_ERROR)==1.

Implements **EVENT::CalorimeterHit** (p. 50).

7.9.3.3 const float ∗ IMPL::CalorimeterHitImpl::getPosition ( ) const [virtual]

Returns the position of the hit in world coordinates.

NULL if information is not stored. Ask collection for flag, only available if bit LCIO.RCHBIT_LONG is set.

Implements **EVENT::CalorimeterHit** (p. 50).

7.9.3.4 EVENT::LCObject ∗ IMPL::CalorimeterHitImpl::getRawHit ( ) const [virtual]

The RawCalorimeterHit.

As in the future there might be other types of raw calorimeter hits the return type is LCObject.

Implements **EVENT::CalorimeterHit** (p. 51).

7.9.3.5 float IMPL::CalorimeterHitImpl::getTime ( ) const [virtual]

Returns the time of the hit in [ns].

Optional, check/set flag(LCIO::RCHBIT_TIME)==1.

Implements **EVENT::CalorimeterHit** (p. 51).

7.9.3.6 int IMPL::CalorimeterHitImpl::getType ( ) const [virtual]

Type of hit.

Mapping of integer types to type names through collection parameters "CalorimeterHitTypeNames" and "-CalorimeterHitTypeValues".

Implements **EVENT::CalorimeterHit** (p. 51).

7.9.3.7 void IMPL::CalorimeterHitImpl::setCellID1 (int *id1*)

Sets the second cell id; Only store if the flag word (bit RCHBIT_ID1) of the collection is set.

Default is to store only cellid0.

7.9.3.8 void IMPL::CalorimeterHitImpl::setEnergyError (float *enerr*)

Sets the error of the hit energy.

Not stored if flag RCHBIT_ENERGY_ERROR isn't set

7.9.3.9 void IMPL::CalorimeterHitImpl::setPosition (const float *pos[3]*)

Sets the position.

Not stored if LCIO.RCHBIT_LONG isn't set.

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

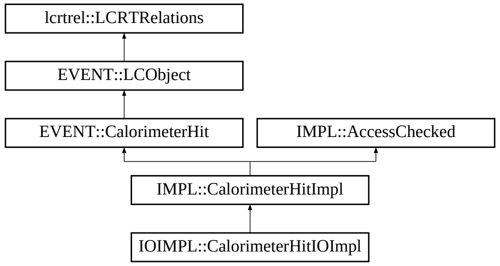
- CalorimeterHitImpl.h
- CalorimeterHitImpl.cc

7.10 IOIMPL::CalorimeterHitIOImpl Class Reference

Adding stuff needed for io (friend declarations, etc.)

`#include <IOIMPL/CalorimeterHitIOImpl.h>`

Inheritance diagram for IOIMPL::CalorimeterHitIOImpl:



Friends

- class **SIO::SIOCalHitHandler**

Additional Inherited Members

7.10.1 Detailed Description

Adding stuff needed for io (friend declarations, etc.)

Author

gaede

Version

Aug 8, 2003

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

- CalorimeterHitIOImpl.h

7.11 lcrtrel_helper::can_call_ext< B > Struct Template Reference

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

- LCRTRelations.h

7.12 lcrtrel_helper::can_call_ext< true > Struct Template Reference

Static Public Member Functions

- static void **check** ()

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

- LCRTRelations.h

7.13 UTIL::CellIDDecoder< T > Class Template Reference

Convenient class for decoding cellIDs from collection parameter LCIO::CellIDEncoding.

```
#include <UTIL/CellIDDecoder.h>
```

Public Member Functions

- **CellIDDecoder** (const std::string &encoder_str)

  Constructor takes encoding string as argument.
- **CellIDDecoder** (const **EVENT::LCCollection** ∗col)

  Constructor reads encoding string from collection parameter LCIO::CellIDEncoding.
- const **BitField64** & **operator()** (const T ∗hit)

  Provides access to the bit fields, e.g.

Static Public Member Functions

- static void **setDefaultEncoding** (const std::string &defaultEncoding)

  This can be used to set the default encoding that is used if no CellIDEncoding parameter is set in the collection, e.g.

Protected Attributes

- **BitField64** ∗ **_b**
- const T ∗ **_oldHit**

Static Protected Attributes

- static std::string ∗ **_defaultEncoding** = new std::string("byte0:8,byte1:8,byte2:8,byte3:8,byte4:8,byte5-:8,byte6:8,byte7:8")

7.13.1 Detailed Description

template<class T>class UTIL::CellIDDecoder< T >

Convenient class for decoding cellIDs from collection parameter LCIO::CellIDEncoding.

See **UTIL::BitField64** (p. 43) for a description of the encoding string.

See Also

BitField64 (p. 43)

Version

Id:

CellIDDecoder.h (p. **??**),v 1.9.16.1 2011-03-04 14:09:07 engels Exp

7.13.2 Member Function Documentation

7.13.2.1 template<class T> const BitField64& UTIL::CellIDDecoder< T >::operator() (const T * *hit*) [inline]

Provides access to the bit fields, e.g.

int layer = myCellIDEncoding(hit)["layer"] ;

7.13.2.2 template<class T> static void UTIL::CellIDDecoder< T >::setDefaultEncoding (const std::string & *defaultEncoding*) [inline], [static]

This can be used to set the default encoding that is used if no CellIDEncoding parameter is set in the collection, e.g.

in older lcio files.

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

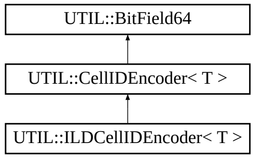
- CellIDDecoder.h

7.14 UTIL::CellIDEncoder< T > Class Template Reference

Specialization for SimTrackerHits that have only one cellID.

`#include <UTIL/CellIDEncoder.h>`

Inheritance diagram for UTIL::CellIDEncoder< T >:



Public Member Functions

- **CellIDEncoder** (const std::string &cellIDEncoding, **EVENT::LCCollection** *col)
  Constructor, sets collection parameter LCIO::CellIDEncoding to the given encoding string.
- void **setCellID** (T *hit)
- void **setCellIDFlag** ()
  Helper method that sets/unsets the proper bit for storing a second cellid word.

Protected Attributes

- **EVENT::LCCollection * _col**

Additional Inherited Members

7.14.1 Detailed Description

template<class T>class UTIL::CellIDEncoder< T >

Specialization for SimTrackerHits that have only one cellID.

Convenient class for encoding cellIDs for various hit objects. It sets the proper collection parameter LCIO::CellID-Encoding and sets the proper flag bit for storing a second cellid if necessary. See **UTIL::BitField64** (p. 43) for a description of the encoding string. Example:

CellIDEncoder<SimCalorimeterHitImpl> cd("i:20,j:20,k:20" ,calVec) ;

for(int j=0;j<NHITS;j++){

SimCalorimeterHitImpl* hit = new SimCalorimeterHitImpl ;

cd["i"] = j ;

cd["j"] = j + 100 ;

cd["k"] = j + 200 ;

cd.setCellID(hit) ;

}

See Also

BitField64 (p. 43)

Version

Id:

CellIDEncoder.h (p. **??**),v 1.6 2008-01-22 14:38:55 gaede Exp

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

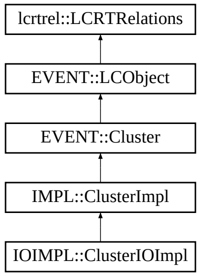
- CellIDEncoder.h

7.15 EVENT::Cluster Class Reference

The **LCIO** (p. 115) cluster.

`#include <pre-generated/EVENT/Cluster.h>`

Inheritance diagram for EVENT::Cluster:



Public Types

- typedef **Cluster lcobject_type**
  *Useful typedef for template programming with **LCIO** (p. 115).*

Public Member Functions

- virtual ∼**Cluster** ()
  Destructor.
- virtual int **getType** () const =0
  Flagword that defines the type of cluster.
- virtual float **getEnergy** () const =0
  Energy of the cluster.
- virtual float **getEnergyError** () const =0
  Returns the error on the energy of the cluster.
- virtual const float ∗ **getPosition** () const =0
  Position of the cluster.
- virtual const **FloatVec** & **getPositionError** () const =0
  Covariance matrix of the position (6 Parameters)
- virtual float **getIPhi** () const =0
  Intrinsic direction of cluster at position: Phi.
- virtual float **getITheta** () const =0
  Intrinsic direction of cluster at position: Theta.
- virtual const **FloatVec** & **getDirectionError** () const =0
  Covariance matrix of the direction (3 Parameters).
- virtual const **FloatVec** & **getShape** () const =0
  Shape parameters - check/set collection parameter ClusterShapeParameters for size and names of parameters.
- virtual const **ParticleIDVec** & **getParticleIDs** () const =0
  The particle Id's sorted by their likelihood.
- virtual const **ClusterVec** & **getClusters** () const =0
  The clusters that have been combined to this cluster.
- virtual const **CalorimeterHitVec** & **getCalorimeterHits** () const =0
  The hits that have been combined to this cluster.
- virtual const **FloatVec** & **getHitContributions** () const =0
  *Returns the energy contribution of the hits Runs parallel to the CalorimeterHitVec from **getCalorimeterHits()** (p. 60)*
- virtual const **FloatVec** & **getSubdetectorEnergies** () const =0
  A vector that holds the energy observed in a particular subdetectors.

Additional Inherited Members

7.15.1 Detailed Description

The **LCIO** (p. 115) cluster.

Author

gaede

Version

Id:

Cluster.aid,v 1.14 2006-08-03 16:53:34 gaede Exp

7.15.2 Member Function Documentation

7.15.2.1 virtual const CalorimeterHitVec& EVENT::Cluster::getCalorimeterHits ( ) const [pure virtual]

The hits that have been combined to this cluster.

Only available if collection flag bit LCIO::CLBIT_HITS==1 and if the **CalorimeterHit** (p. 49) objects have not been saved with LCIO::RCHBIT_NO_PTR==1.

See Also

CalorimeterHit (p. 49)

Implemented in **IMPL::ClusterImpl** (p. 62).

Referenced by SIO::SIOClusterHandler::write().

7.15.2.2 virtual const FloatVec& EVENT::Cluster::getDirectionError ( ) const [pure virtual]

Covariance matrix of the direction (3 Parameters).

Stored as lower triangle matrix where the order of parameters is: iPhi, iTheta. So we have cov(iPhi,iPhi), cov(iPhi, iTheta), cov(iTheta, iTheta).

Implemented in **IMPL::ClusterImpl** (p. 62).

Referenced by SIO::SIOClusterHandler::write().

7.15.2.3 virtual float EVENT::Cluster::getIPhi ( ) const [pure virtual]

Intrinsic direction of cluster at position: Phi.

Not to be confused with direction cluster is seen from IP.

Implemented in **IMPL::ClusterImpl** (p. 62).

Referenced by SIO::SIOClusterHandler::write().

7.15.2.4 virtual float EVENT::Cluster::getITheta ( ) const [pure virtual]

Intrinsic direction of cluster at position: Theta.

Not to be confused with direction cluster is seen from IP.

Implemented in **IMPL::ClusterImpl** (p. 62).

Referenced by SIO::SIOClusterHandler::write().

7.15.2.5 virtual const ParticleIDVec& EVENT::Cluster::getParticleIDs ( ) const [pure virtual]

The particle Id's sorted by their likelihood.

See Also

ParticleID (p. 181)

Implemented in **IMPL::ClusterImpl** (p. 63).

Referenced by UTIL::LCTOOLS::printClusters(), and SIO::SIOClusterHandler::write().

7.15.2.6 virtual const FloatVec& EVENT::Cluster::getSubdetectorEnergies ( ) const [pure virtual]

A vector that holds the energy observed in a particular subdetectors.

The mapping of indices to subdetectors is implementation dependent. To be used as convenient information or if hits are not stored in the data set, e.g. DST or FastMC. Check/set collection parameter ClusterSubdetectorNames for decoding the indices of the array.

Implemented in **IMPL::ClusterImpl** (p. 64).

Referenced by SIO::SIOClusterHandler::write().

7.15.2.7 virtual int EVENT::Cluster::getType ( ) const [pure virtual]

Flagword that defines the type of cluster.

Bits 0-15 can be used to denote the subdetectors that have contributed hits to the cluster. For the definition of the bits check/Set the collection variables ClusterTypeBitNames and ClusterTypeBitIndices. Bits 16-31 are used internally.

Implemented in **IMPL::ClusterImpl** (p. 64).

Referenced by SIO::SIOClusterHandler::write().

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

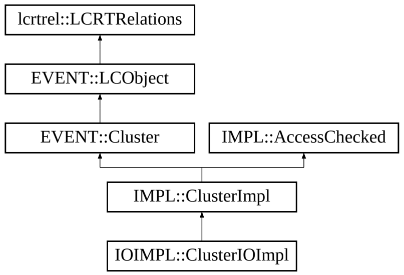
- Cluster.h

7.16 IMPL::ClusterImpl Class Reference

Implementation of Cluster.

`#include <IMPL/ClusterImpl.h>`

Inheritance diagram for IMPL::ClusterImpl:



Public Member Functions

- **ClusterImpl** ()
  Default constructor, initializes values to 0.
- virtual ∼**ClusterImpl** ()
  Destructor.
- virtual int **id** () const
  Returns an object id for internal (debugging) use in LCIO.
- virtual int **getType** () const

  Flagword that defines the type of cluster.
- virtual float **getEnergy** () const
  Energy of the cluster.
- virtual float **getEnergyError** () const
  Returns the error on the energy of the cluster.
- virtual const float ∗ **getPosition** () const
  Position of the cluster.
- virtual const **EVENT::FloatVec** & **getPositionError** () const
  Covariance matrix of the position (6 Parameters)
- virtual float **getITheta** () const
  Intrinsic direction of cluster at position: Theta.
- virtual float **getIPhi** () const
  Intrinsic direction of cluster at position: Phi.
- virtual const **EVENT::FloatVec** & **getDirectionError** () const
  Covariance matrix of the direction (3 Parameters)
- virtual const **EVENT::FloatVec** & **getShape** () const
  Shape parameters - check/set collection parameter ClusterShapeParameters for size and names of parameters.
- virtual const **EVENT::ParticleIDVec** & **getParticleIDs** () const
  Type hypotheses: 3 Parameters: compatible with EM, HAD, muon cluster.
- virtual const **EVENT::ClusterVec** & **getClusters** () const
  The clusters that have been combined to this cluster.
- virtual const **EVENT::CalorimeterHitVec** & **getCalorimeterHits** () const
  The hits that have been combined to this cluster.
- virtual const **EVENT::FloatVec** & **getHitContributions** () const
  *Returns the energy contribution of the hits Runs parallel to the CalorimeterHitVec from **getCalorimeterHits()** (p. 62)*
- virtual const **EVENT::FloatVec** & **getSubdetectorEnergies** () const
  A vector that holds the energy observed in a particular subdetectors.
- void **setTypeBit** (int index, bool val=true)
- void **setEnergy** (float energy)
- void **setEnergyError** (float energyError)
- void **setPosition** (const float ∗position)
- void **setPositionError** (const **EVENT::FloatVec** &errpos)
- void **setPositionError** (const float ∗errpos)
- void **setITheta** (float theta)
- void **setIPhi** (float phi)
- void **setDirectionError** (const **EVENT::FloatVec** &errdir)
- void **setDirectionError** (const float ∗errdir)
- void **setShape** (const **EVENT::FloatVec** &shape)
- void **addParticleID** (**EVENT::ParticleID** ∗pid)
- void **addCluster** (**EVENT::Cluster** ∗cluster)
- void **addHit** (**EVENT::CalorimeterHit** ∗hit, float contribution)
- **EVENT::FloatVec** & **subdetectorEnergies** ()
  To be used for modifying the subdetector energies, e.g.

Protected Member Functions

- void **setType** (int type)

Protected Attributes

- std::bitset< 32 > **_type**
- float **_energy**
- float **_energyError**
- float **_position** [3]
- **EVENT::FloatVec _errpos**
- float **_theta**
- float **_phi**
- **EVENT::FloatVec _errdir**
- **EVENT::FloatVec _shape**
- **EVENT::ParticleIDVec _pid**
- **EVENT::ClusterVec _clusters**
- **EVENT::CalorimeterHitVec _hits**
- **EVENT::FloatVec _weights**
- **EVENT::FloatVec _subdetectorEnergies**

Additional Inherited Members

7.16.1 Detailed Description

Implementation of Cluster.

See Also

Cluster

Author

gaede

Version

Id:

ClusterImpl.h (p. **??**),v 1.13 2006-08-04 16:52:46 gaede Exp

7.16.2 Member Function Documentation

7.16.2.1 const EVENT::ParticleIDVec & IMPL::ClusterImpl::getParticleIDs ( ) const [virtual]

Type hypotheses: 3 Parameters: compatible with EM, HAD, muon cluster.

The particle Id's sorted by their probability.

See Also

ParticleID

Implements **EVENT::Cluster** (p. 60).

7.16.2.2 const EVENT::FloatVec & IMPL::ClusterImpl::getSubdetectorEnergies ( ) const [virtual]

A vector that holds the energy observed in a particular subdetectors.

The mapping of indices to subdetectors is implementation dependent. To be used as convenient information or if hits are not stored in the data set, e.g. DST or FastMC. Check/set collection parameter ClusterSubdetectorNames for decoding the indices of the array.

Implements **EVENT::Cluster** (p. 61).

7.16.2.3 int IMPL::ClusterImpl::getType ( ) const [virtual]

Flagword that defines the type of cluster.

Bits 0-15 can be used to denote the subdetectors that have contributed hits to the cluster. The definition of the bits has to be done elsewhere, e.g. in the run header. Bits 16-31 are used internally.

Implements **EVENT::Cluster** (p. 61).

7.16.2.4 EVENT::FloatVec & IMPL::ClusterImpl::subdetectorEnergies ( )

To be used for modifying the subdetector energies, e.g.

clu->**subdetectorEnergies()** (p. 64).resize(3) ; clu->**subdetectorEnergies()** (p. 64)[0] = 123.4567 ; clu->**subdetectorEnergies()** (p. 64)[1] = 12.34567 ; clu->**subdetectorEnergies()** (p. 64)[2] = 1.234567 ;

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

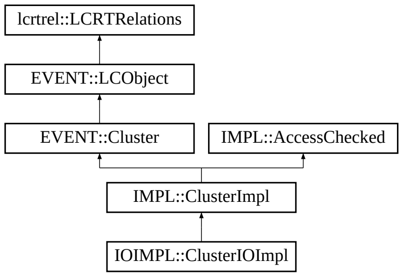
- ClusterImpl.h
- ClusterImpl.cc

7.17 IOIMPL::ClusterIOImpl Class Reference

Adding stuff needed for io (friend declarations, etc.)

`#include <IOIMPL/ClusterIOImpl.h>`

Inheritance diagram for IOIMPL::ClusterIOImpl:



Friends

- class **SIO::SIOClusterHandler**

Additional Inherited Members

7.17.1 Detailed Description

Adding stuff needed for io (friend declarations, etc.)

Author

gaede

Version

Mar 15, 2004

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

- ClusterIOImpl.h

7.18 UTIL::CollectionParameterMap Class Reference

Helper class for setting and retrieving an std::map<string,int> as collection parameters.

```
#include <UTIL/CollectionParameterMap.h>
```

Public Types

- typedef std::map< std::string,
  int > **map_type**

Public Member Functions

- **CollectionParameterMap** (const std::string &keyName, const std::string &valueName, **LCCollection** ∗col)
  Create ***CollectionParameterMap*** *(p. 65) for given collection - read the collection parameters keyName and valueName if they exist.*
- **CollectionParameterMap** (const std::string &keyName, const std::string &valueName, const **LCCollection** ∗col)
  Create ***CollectionParameterMap*** *(p. 65) for given collection - read the collection parameters keyName and valueName if they exist.*
- ∼**CollectionParameterMap** ()
  Update the collection parameters keyName and valueName when going out of scope.
- map_type & **map** ()
  The std::map< std::string, int >.

Protected Member Functions

- void **init** (const **LCCollection** ∗col)

Protected Attributes

- std::string **_keyName**
- std::string **_valueName**
- **LCCollection** ∗ **_col**
- map_type **_map**

7.18.1 Detailed Description

Helper class for setting and retrieving an std::map<string,int> as collection parameters.

Depending on which constructor was used the collection parameters will be updated if an object of this class goes out of scope. The corresponding collection parameters will hold the current state of the map. If the a const LCCollection∗ was specified (read use case) no parameters will be updated.

See Also

LCCollection::getParameters()
LCParameters

Author

F.Gaede, DESY

Version

Id:

CollectionParameterMap.h (p. **??**),v 1.1 2008-05-30 13:26:16 gaede Exp

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

- CollectionParameterMap.h
- CollectionParameterMap.cc

7.19 lcrtrel_helper::CreationPtrInit< T > Struct Template Reference

Factory for objects of type T.

```
#include <LCRTRelations.h>
```

Static Public Member Functions

- static void ∗ **init** ()

7.19.1 Detailed Description

template<class T>struct lcrtrel_helper::CreationPtrInit< T >

Factory for objects of type T.

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

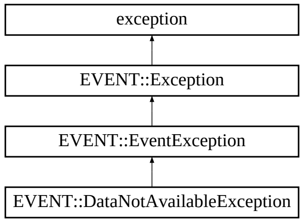
- LCRTRelations.h

7.20 EVENT::DataNotAvailableException Class Reference

EventException (p. 69) used for data not available.

```
#include <Exceptions.h>
```

Inheritance diagram for EVENT::DataNotAvailableException:



Public Member Functions

- **DataNotAvailableException** (std::string text)

Additional Inherited Members

7.20.1 Detailed Description

EventException (p. 69) used for data not available.

Author

gaede

Version

Jun 5, 2003

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

- Exceptions.h

7.21 lcrtrel_helper::DeleteElements< T > Struct Template Reference

Delete function for containers of owned objects.

```
#include <LCRTRelations.h>
```

Static Public Member Functions

- static void **clean** (void ∗v)

7.21.1 Detailed Description

template< class T > struct lcrtrel_helper::DeleteElements< T >

Delete function for containers of owned objects.

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

- LCRTRelations.h

7.22 lcrtrel_helper::DeletePtr< T > Struct Template Reference

Delete function for pointers w/ ownership.

```
#include <LCRTRelations.h>
```

Static Public Member Functions

- static void **clean** (void ∗v)

7.22.1 Detailed Description

template< class T > struct lcrtrel_helper::DeletePtr< T >

Delete function for pointers w/ ownership.

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

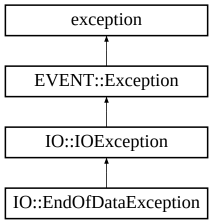
- LCRTRelations.h

7.23 IO::EndOfDataException Class Reference

EndOfDataException (p. 68) for signaling the end of a data stream.

```
#include <Exceptions.h>
```

Inheritance diagram for IO::EndOfDataException:



Public Member Functions

- **EndOfDataException** (std::string text)

Additional Inherited Members

7.23.1 Detailed Description

EndOfDataException (p. 68) for signaling the end of a data stream.

Author

gaede

Version

Jun 5, 2003

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

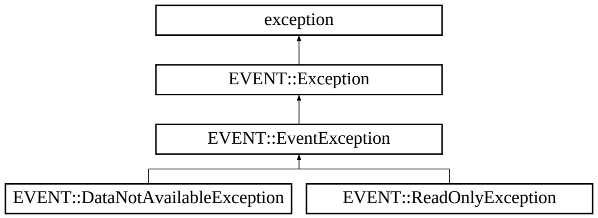
- Exceptions.h

7.24 EVENT::EventException Class Reference

EventException (p. 69) used for errors accessing the event data.

`#include <Exceptions.h>`

Inheritance diagram for EVENT::EventException:



Public Member Functions

- **EventException** (std::string text)

Additional Inherited Members

7.24.1 Detailed Description

EventException (p. 69) used for errors accessing the event data.

Author

gaede

Version

Apr 30, 2003

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

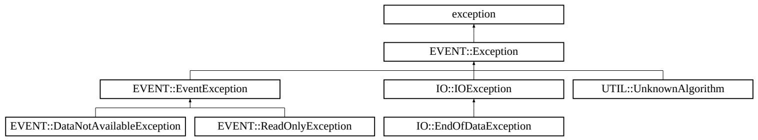
- Exceptions.h

7.25 EVENT::Exception Class Reference

Base exception class for **LCIO** (p. 115) - all other exceptions extend this.

`#include <Exceptions.h>`

Inheritance diagram for EVENT::Exception:



Public Member Functions

- **Exception** (const std::string &text)
- virtual const char $*$ **what** () const throw ()

Protected Attributes

- std::string **message**

7.25.1 Detailed Description

Base exception class for **LCIO** (p. 115) - all other exceptions extend this.

Author

gaede

Version

Apr 30, 2003

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

- Exceptions.h

7.26 lcrtrel_helper::FromRelation< U > Struct Template Reference

Helper class for relations.

`#include <LCRTRelations.h>`

7.26.1 Detailed Description

template< class U > struct lcrtrel_helper::FromRelation< U >

Helper class for relations.

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

- LCRTRelations.h

7.27 HEPEVTIMPL::HEPEVT Class Reference

Static Public Member Functions

- static void **fromHepEvt** (**EVENT::LCEvent** ∗evt, const char ∗mcColName=0)

    StdHep fortran interface to read from or write to the COMMON /HEPEVT/.
- static void **toHepEvt** (const **EVENT::LCEvent** ∗evt, const char ∗mcColName=0)

7.27.1 Member Function Documentation

7.27.1.1 void HEPEVTIMPL::HEPEVT::fromHepEvt (EVENT::LCEvent ∗ *evt*, const char ∗ *mcColName* = 0) [static]

StdHep fortran interface to read from or write to the COMMON /HEPEVT/.

fromHepEvt reads from COMMON /HEPEVT/. toHepEvt fills COMMON /HEPEVT/.

References IMPL::MCParticleImpl::addParent(), IMPL::LCEventImpl::getCollection(), EVENT::LCCollection::getElementAt(), IMPL::LCCollectionVec::getElementAt(), EVENT::LCCollection::getNumberOfElements(), IMPL::MCParticleImpl::setCharge(), IMPL::MCParticleImpl::setGeneratorStatus(), IMPL::MCParticleImpl::setMass(), IMPL::MCParticleImpl::setMomentum(), IMPL::MCParticleImpl::setPDG(), IMPL::MCParticleImpl::setSimulatorStatus(), and IMPL::MCParticleImpl::setVertex().

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

- HEPEVT.h
- HEPEVT.cc

7.28 hepevt1_t Struct Reference

Public Attributes

- PTRTYPE **mcpointerv** [NMXHEP]
- float **mcchargev** [NMXHEP]

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

- hepevt1.h

7.29 hepevt_t Struct Reference

Public Attributes

- int **nevhep**
- int **nhep**
- int **isthep** [NMXHEP]
- int **idhep** [NMXHEP]
- int **jmohep** [NMXHEP][2]
- int **jdahep** [NMXHEP][2]
- double **phep** [NMXHEP][5]
- double **vhep** [NMXHEP][4]

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

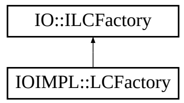
- hepevt0.h

7.30 IO::ILCFactory Class Reference

Factory to create **LCWriter** (p. 166) and Reader object for the known data formats, e.g.

`#include <pre-generated/IO/ILCFactory.h>`

Inheritance diagram for IO::ILCFactory:



Public Member Functions

- virtual ∼**ILCFactory** ()

    Destructor.
- virtual **LCWriter** ∗ **createLCWriter** ()=0

    Returns the instance of the factory.
- virtual **LCReader** ∗ **createLCReader** (int readerFlag=0)=0

    *Creates an **LCReader** (p. 130) object for the current persistency type.*

7.30.1 Detailed Description

Factory to create **LCWriter** (p. 166) and Reader object for the known data formats, e.g.

SIO (p. 30). The implementation (LCFactory) will be a singleton class that knows all concrete reader/writer implementations.

Author

gaede

Version

Mar 7, 2003

7.30.2 Member Function Documentation

7.30.2.1 virtual LCWriter∗ IO::ILCFactory::createLCWriter ( ) [pure virtual]

Returns the instance of the factory.

In the future we need to specify the data format here... This doesn't work for java as an interface can't have static members ...Creates an **LCWriter** (p. 166) object for the current persistency type.

Implemented in **IOIMPL::LCFactory** (p. 101).

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

- ILCFactory.h

7.31 UTIL::ILDCellID0 Struct Reference

Helper class to define constants for the canonical encoding of the CellID0 in tracking sub detectors in ILD (or ILD like detectors).

```
#include <UTIL/ILDConf.h>
```

Static Public Attributes

- static std::string **encoder_string** = "subdet:5,side:-2,layer:9,module:8,sensor:8"
  The canonical encoding string to be used for writing the CellID0 with the CellIDEncoder<T>:
- static const unsigned **subdet** = 0
  ***Index** (p. 77) of the field subdet in encoder_string - sub detector Id as defined in DetID, e.g.*
- static const unsigned **side** = 1
  ***Index** (p. 77) of the field side in encoder_string - one of DetID::bwd, DetID::barrel, DetID::fwd.*
- static const unsigned **layer** = 2
  ***Index** (p. 77) of the field layer in encoder_string - local sub detector layer ID starting from 0 going away from the origin (IP).*
- static const unsigned **module** = 3
  ***Index** (p. 77) of the field module in encoder_string - module ID as defined for the given subdetetor.*
- static const unsigned **sensor** = 4
  ***Index** (p. 77) of the field sensor in encoder_string - sensor ID as defined for the given subdetetor module.*

7.31.1 Detailed Description

Helper class to define constants for the canonical encoding of the CellID0 in tracking sub detectors in ILD (or ILD like detectors).

F.Gaede, DESY, 2011

Version

$Id:$

7.31.2 Member Data Documentation

7.31.2.1 std::string UTIL::ILDCellID0::encoder_string = "subdet:5,side:-2,layer:9,module:8,sensor:8" [static]

The canonical encoding string to be used for writing the CellID0 with the CellIDEncoder<T>:

```
"subdet:5,side:-2,layer:9,module:8:sensor:8"
```

Use the CellIdDecoder to access the elements:

```
CellIDDecoder<TrackerHit> idDec( trkHitCol ) ;
 // ...
 TrackerHit* hit = (TrackerHit*) trkHitCol.getElementAt( i ) ;

 int layer  = idDec( hit )[ ILDCellID0::layer (p.73) ] ;
```

7.31.2.2 const unsigned UTIL::ILDCellID0::module = 3 [static]

Index (p. 77) of the field module in encoder_string - module ID as defined for the given subdetetor.

7.31.2.3 const unsigned UTIL::ILDCellID0::sensor = 4 [static]

Index (p. 77) of the field sensor in encoder_string - sensor ID as defined for the given subdetetor module.

7.31.2.4 const unsigned UTIL::ILDCellID0::subdet = 0 [static]

Index (p. 77) of the field subdet in encoder_string - sub detector Id as defined in DetID, e.g.

DetID::VXD or DetID::FTD.

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

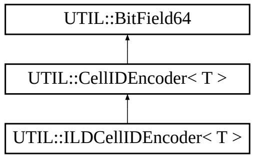
- ILDConf.h
- ILDConf.cc

7.32 UTIL::ILDCellIDEncoder< T > Class Template Reference

Special wrapper to the CellIDEncoder<T> that enforces cellID0 to use the encoder string defined in **ILDCellID0::encoder_string** (p. 73).

```
#include <UTIL/ILDConf.h>
```

Inheritance diagram for UTIL::ILDCellIDEncoder< T >:



Public Member Functions

- **ILDCellIDEncoder** (**EVENT::LCCollection** ∗col)
  *Constructor for using the canonical cellID0 as defined in **ILDCellID0::encoder_string** (p. 73) - cellID1 will not be used.*
- **ILDCellIDEncoder** (const std::string &cellIDEncoding, **EVENT::LCCollection** ∗col)
  *Constructor that prepends given **ILDCellID0::encoder_string** (p. 73) to the given encoding string.*

Additional Inherited Members

7.32.1 Detailed Description

template<class T>class UTIL::ILDCellIDEncoder< T >

Special wrapper to the CellIDEncoder<T> that enforces cellID0 to use the encoder string defined in **ILDCellID0::encoder_string** (p. 73).

7.32.2 Constructor & Destructor Documentation

7.32.2.1 template< class T > UTIL::ILDCellIDEncoder< T >::ILDCellIDEncoder (const std::string & *cellIDEncoding*, EVENT::LCCollection * *col*) [inline]

Constructor that prepends given **ILDCellID0::encoder_string** (p. 73) to the given encoding string.

Throws an Exception if called with an empty string.

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

- ILDConf.h

7.33 UTIL::ILDDetID Struct Reference

Define integer constants for identifying ILD sub detectors.

`#include <UTIL/ILDConf.h>`

Static Public Attributes

- static const int **NOTUSED** = 0
- static const int **VXD** = 1
- static const int **SIT** = 2
- static const int **FTD** = 3
- static const int **TPC** = 4
- static const int **SET** = 5
- static const int **ETD** = 6
- static const int **ECAL** = 20
- static const int **ECAL_PLUG** = 21
- static const int **HCAL** = 22
- static const int **HCAL_RING** = 23
- static const int **LCAL** = 24
- static const int **BCAL** = 25
- static const int **LHCAL** = 26
- static const int **YOKE** = 27
- static const int **COIL** = 28
- static const int **ECAL_ENDCAP** = 29
- static const int **HCAL_ENDCAP** = 30
- static const int **YOKE_ENDCAP** = 31
- static const int **bwd** = - 1
- static const int **barrel** = 0
- static const int **fwd** = 1

7.33.1 Detailed Description

Define integer constants for identifying ILD sub detectors.

Typically used in CellIDs, for example together with ILD::CellID0::encoder_string.

F.Gaede, DESY, 2011

Version

$Id:$

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

- ILDConf.h
- ILDConf.cc

7.34 UTIL::ILDTrkHitQualityBit Struct Reference

Define integer constants for bits of the the lcio tracker hit quality used in the context of ILD reconstruction code.

`#include <UTIL/ILDConf.h>`

Static Public Attributes

- static const int **USED_IN_FIT** = 30
- static const int **USED_IN_TRACK** = 29
- static const int **DOUBLE_HIT_CANDIDATE** = 28
- static const int **GOOD** = 27

7.34.1 Detailed Description

Define integer constants for bits of the the lcio tracker hit quality used in the context of ILD reconstruction code.

F.Gaede, DESY, 2012

Version

$Id:$

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

- ILDConf.h
- ILDConf.cc

7.35 UTIL::ILDTrkHitTypeBit Struct Reference

Define integer constants for bits of the the lcio tracker hit type used in the context of ILD reconstruction code.

`#include <UTIL/ILDConf.h>`

Static Public Attributes

- static const int **COMPOSITE_SPACEPOINT** = 30
- static const int **ONE_DIMENSIONAL** = 29

7.35.1 Detailed Description

Define integer constants for bits of the the lcio tracker hit type used in the context of ILD reconstruction code.

S.J. Aplin, DESY, 2012

Version

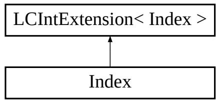
$Id:$

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

- ILDConf.h
- ILDConf.cc

7.36 Index Struct Reference

Inheritance diagram for Index:

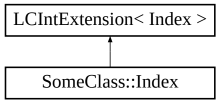


The documentation for this struct was generated from the following file:

- lcrtrelation.cc

7.37 SomeClass::Index Struct Reference

Inheritance diagram for SomeClass::Index:



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

- test_RTextension.cc

7.38 UTIL::IndexMap Class Reference

utility class to manage indexes according to Collection Parameters

```
#include <UTIL/IndexMap.h>
```

Public Member Functions

- **IndexMap** (const **EVENT::LCCollection** ∗col, const std::string &key1, const std::string &key2)

  *Constructor: takes as arguments the collection where the parameters are stored and the parameter names to manage Before you loop over a collection create an **IndexMap** (p. 77) object with the keys to manage.*
- int **encode** (const std::string name)

  encode string to index (e.g.
- const std::string & **decode** (int type) const

  decode int to string (e.g.

Protected Attributes

- const **EVENT::LCCollection** ∗ **_col**
- const std::string **_key1**
- const std::string **_key2**
- **EVENT::StringVec _strvec**
- **EVENT::IntVec _intvec**

7.38.1 Detailed Description

utility class to manage indexes according to Collection Parameters

EXP: UNDER DEVELOPMENT!!! - Don't use this class yet

Author

engels

Version

Id:

IndexMap.h (p. **??**),v 1.2 2006-09-21 06:10:38 gaede Exp

7.38.2 Constructor & Destructor Documentation

7.38.2.1 UTIL::IndexMap::IndexMap (const EVENT::LCCollection ∗ *col*, const std::string & *key1*, const std::string & *key2*)

Constructor: takes as arguments the collection where the parameters are stored and the parameter names to manage Before you loop over a collection create an **IndexMap** (p. 77) object with the keys to manage.

Example:

LCCollection∗ col = evt->getCollection("Vertices") ;

IndexMap (p. 77) imvtx(col, "AlgorithmNames", "AlgorithmTypes") ;

Loop over Collection....

Vertex∗ v = dynamic_cast<Vertex∗>(col->getElementAt(i)) ;

cout << "vertex " << i << has been created with " << imvtx.decode(v->getAlgorithmType()) << endl;

End Loop

For using the encoding use similar process:

IndexMap (p. 77) imvtx(col, "AlgorithmNames", "AlgorithmTypes") ;

Loop for creating vertices...

VertexImpl∗ v = new VertexImpl ;

v->setAlgorithmType(imvtx.encode("ZvTop"));

// If the parameter "ZvTop" or even the collection parameters "AlgorithmNames" and "AlgorithmTypes" do not exist // they are automatically added and ZvTop gets a corresponding new AlgorithmType which is also automatically added

End Loop

References EVENT::LCParameters::getIntVals(), EVENT::LCCollection::getParameters(), and EVENT::LCParameters::getStringVals().

7.38.3 Member Function Documentation

7.38.3.1 const std::string & UTIL::IndexMap::decode (int *type*) const

decode int to string (e.g.

cout << "vertex has been created with " << imvtx.decode(vtx->getAlgorithmType()) << endl;

7.38.3.2 int UTIL::IndexMap::encode (const std::string *name*)

encode string to index (e.g.

vertex->setAlgorithmType(imvtx.enconde("ZvTop"));

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

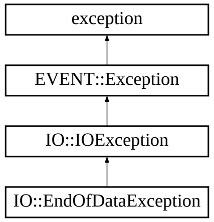
- IndexMap.h
- IndexMap.cc

7.39 IO::IOException Class Reference

IOException (p. 79) used for reading/writing errors.

`#include <Exceptions.h>`

Inheritance diagram for IO::IOException:



Public Member Functions

- **IOException** (std::string text)

Additional Inherited Members

7.39.1 Detailed Description

IOException (p. 79) used for reading/writing errors.

Author

gaede

Version

Apr 30, 2003

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

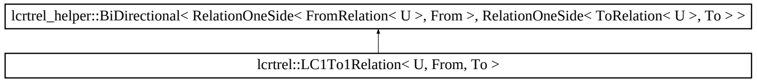
- Exceptions.h

7.40 lcrtrel::LC1To1Relation< U, From, To > Struct Template Reference

One to one relation between two objects of type From and To.

`#include <LCRTRelations.h>`

Inheritance diagram for lcrtrel::LC1To1Relation< U, From, To >:



Additional Inherited Members

7.40.1 Detailed Description

template<class U, class From, class To>struct lcrtrel::LC1To1Relation< U, From, To >

One to one relation between two objects of type From and To.

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

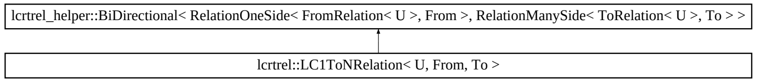
- LCRTRelations.h

7.41 lcrtrel::LC1ToNRelation< U, From, To > Struct Template Reference

One to many relation between one object of type From to many objects of type To.

`#include <LCRTRelations.h>`

Inheritance diagram for lcrtrel::LC1ToNRelation< U, From, To >:



Additional Inherited Members

7.41.1 Detailed Description

template<class U, class From, class To>struct lcrtrel::LC1ToNRelation< U, From, To >

One to many relation between one object of type From to many objects of type To.

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

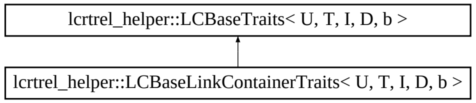
- LCRTRelations.h

7.42 lcrtrel_helper::LCBaseLinkContainerTraits< U, T, I, D, b > Struct Template Reference

Base class for all containers of extensions and relations, vectors, lists,...

`#include <LCRTRelations.h>`

Inheritance diagram for lcrtrel_helper::LCBaseLinkContainerTraits< U, T, I, D, b >:



Public Types

- typedef **LCBaseTraits**< U, T, I, D, b > **base**
- typedef T ∗ **ext_type**
- typedef const T ∗ **rel_type**
- typedef T::value_type **obj_ptr**
- typedef T::iterator **iterator**
- typedef T::const_iterator **const_iterator**

Static Public Attributes

- static const bool **is_container** =true

Additional Inherited Members

7.42.1 Detailed Description

template<class U, class T, class I = CreationPtrInit<T>, class D = DeletePtr<T>, bool b = 1>struct lcrtrel_helper::LCBaseLinkContainerTraits< U, T, I, D, b >

Base class for all containers of extensions and relations, vectors, lists,...

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

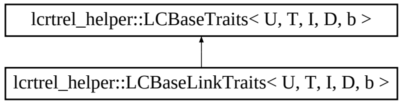
- LCRTRelations.h

7.43 lcrtrel_helper::LCBaseLinkTraits< U, T, I, D, b > Struct Template Reference

Base class for all extensions and relations of single objects.

`#include <LCRTRelations.h>`

Inheritance diagram for lcrtrel_helper::LCBaseLinkTraits< U, T, I, D, b >:



Public Types

- typedef **LCBaseTraits**< U, T, I, D, b > **base**
- typedef T ∗& **ext_type**
- typedef T ∗ **rel_type**
- typedef **base::ptr obj_ptr**

Static Public Attributes

- static const bool **is_container** =false

Additional Inherited Members

7.43.1 Detailed Description

template<class U, class T, class I = SimplePtrInit, class D = NoDelete, bool b = 1>struct lcrtrel_helper::LCBaseLinkTraits< U, T, I, D, b >

Base class for all extensions and relations of single objects.

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

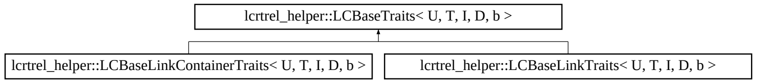
- LCRTRelations.h

7.44 lcrtrel_helper::LCBaseTraits< U, T, I, D, b > Struct Template Reference

Vector of pointers to extension obbjects.

`#include <LCRTRelations.h>`

Inheritance diagram for lcrtrel_helper::LCBaseTraits< U, T, I, D, b >:



Public Types

- typedef T ∗ **ptr**

  base pointer type
- typedef U **tag**

Static Public Member Functions

- static void **clean** (void ∗v)
- static **ptr init** ()
- static DeleteFPtr **deletePtr** ()

Static Public Attributes

- static const int **allowed_to_call_ext** = b

7.44.1 Detailed Description

template<class U, class T, class I, class D, bool b>struct lcrtrel_helper::LCBaseTraits< U, T, I, D, b >

Vector of pointers to extension obbjects.

Base class for all extensions and relations

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

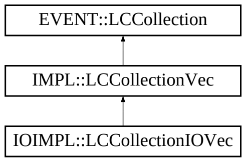
- LCRTRelations.h

7.45 EVENT::LCCollection Class Reference

The generic collection used in **LCIO** (p. 115).

`#include <pre-generated/EVENT/LCCollection.h>`

Inheritance diagram for EVENT::LCCollection:



Public Member Functions

- virtual ∼**LCCollection** ()
  Destructor.
- virtual int **getNumberOfElements** () const =0
  Returns the number of elements in the collection.
- virtual const std::string & **getTypeName** () const =0
  *Returns the type name of the collection - valid names are defined in **LCIO** (p. 115).*
- virtual **LCObject** ∗ **getElementAt** (int index) const =0
  Returns pointer to element at index - no range check, use getNumberOfEntries().
- virtual int **getFlag** () const =0
  Returns flag word for collection.
- virtual bool **isTransient** () const =0
  True if collection is transient, i.e.
- virtual bool **isDefault** () const =0
  True if collection is the default collection for the given type.
- virtual bool **isSubset** () const =0
  True if the collection holds a subset of objects from other collections.
- virtual void **addElement** (**LCObject** ∗obj)=0 throw (ReadOnlyException, std::exception)
  Adds the given element to (end of) the collection.
- virtual void **removeElementAt** (int i)=0 throw (ReadOnlyException, std::exception)
  Removes the i-th element from the collection.
- virtual void **setFlag** (int flag)=0
  Set the flag word.
- virtual const **LCParameters** & **getParameters** () const =0

  Parameters defined for this collection.
- virtual **LCParameters** & **parameters** ()=0
  Parameters defined for this collection.

Static Public Attributes

- static const int **BITTransient** = 16
  Transient bit in flag word.
- static const int **BITDefault** = 17
- static const int **BITSubset** = 18

7.45.1 Detailed Description

The generic collection used in **LCIO** (p. 115).

Holds arbitrary objects of type **LCObject** (p. 123).

Author

gaede

Version

Id:

LCCollection.aid,v 1.11 2005-03-02 16:22:58 gaede Exp

See Also

LCObject (p. 123)
LCIO (p. 115)

7.45.2 Member Function Documentation

7.45.2.1 virtual void EVENT::LCCollection::addElement (LCObject ∗ *obj*) throw ReadOnlyException, std::exception) [pure virtual]

Adds the given element to (end of) the collection.

Throws an exception if the collection (event) is 'read only'.

Exceptions

ReadOnlyException *(p. 192)*	

Implemented in **IMPL::LCCollectionVec** (p. 89).

7.45.2.2 virtual int EVENT::LCCollection::getFlag ( ) const [pure virtual]

Returns flag word for collection.

Bits 16-31 are reserved for **LCIO** (p. 115) Depending on the object type stored they have a special meaning, e.g. for SimCalorimeterHits:

CHBIT_LONG = 31 - store position

CHBIT_BARREL = 30 - endcap or barrel

CHBIT_ID1 = 29 - cellid1 is sored

CHBIT_PDG = 28 - store pdg of secondaries

Bit 16 is used to flag collection as transient

Bit 17 is used to flag collection as default

Bit 18 is used to flag collection as subset

Bits 0-15 are subdetector/user specific.

See Also

isTransient() (p. 86)

Implemented in **IMPL::LCCollectionVec** (p. 89).

Referenced by SIO::SIOVertexHandler::init(), SIO::SIOLCGenericObjectHandler::init(), SIO::SIOObjectHandler::init(), UTIL::LCTOOLS::printCalorimeterHits(), UTIL::LCTOOLS::printClusters(), UTIL::LCTOOLS::printLCFloatVecs(), UTIL::LCTOOLS::printLCGenericObjects(), UTIL::LCTOOLS::printLCIntVecs(), UTIL::LCTOOLS::printLCStrVecs(), UTIL::LCTOOLS::printMCParticles(), UTIL::LCTOOLS::printRawCalorimeterHits(), UTIL::LCTOOLS::printReconstructedParticles(), UTIL::LCTOOLS::printRelation(), UTIL::LCTOOLS::printSimCalorimeterHits(), UTIL::LCTOOLS::printSimTrackerHits(), UTIL::LCTOOLS::printTPCHits(), UTIL::LCTOOLS::printTrackerData(), UTIL::LCTOOLS::printTrackerHitPlane(), UTIL::LCTOOLS::printTrackerHits(), UTIL::LCTOOLS::printTrackerHitZCylinder(), UTIL::LCTOOLS::printTrackerPulse(), UTIL::LCTOOLS::printTrackerRawData(), UTIL::LCTOOLS::printTracks(), and UTIL::CellIDEncoder< T >::setCellIDFlag().

7.45.2.3 virtual int EVENT::LCCollection::getNumberOfElements ( ) const [pure virtual]

Returns the number of elements in the collection.

Same as size().

Implemented in **IMPL::LCCollectionVec** (p. 88).

Referenced by UTIL::LCTOOLS::dumpEvent(), HEPEVTIMPL::HEPEVT::fromHepEvt(), SIO::SIOVertexHandler::init(), SIO::SIOLCGenericObjectHandler::init(), UTIL::LCIterator< T >::LCIterator(), UTIL::LCTOOLS::printCalorimeterHits(), UTIL::LCTOOLS::printClusters(), UTIL::LCTOOLS::printLCFloatVecs(), UTIL::LCTOOLS::printLCGenericObjects(), UTIL::LCTOOLS::printLCIntVecs(), UTIL::LCTOOLS::printLCStrVecs(), UTIL::LCTOOLS::printMCParticles(), UTIL::LCTOOLS::printRawCalorimeterHits(), UTIL::LCTOOLS::printReconstructedParticles(), UTIL::LCTOOLS::printRelation(), UTIL::LCTOOLS::printSimCalorimeterHits(), UTIL::LCTOOLS::printSimTrackerHits(), UTIL::LCTOOLS::printTPCHits(), UTIL::LCTOOLS::printTrackerData(), UTIL::LCTOOLS::printTrackerHitPlane(), UTIL::LCTOOLS::printTrackerHits(), UTIL::LCTOOLS::printTrackerHitZCylinder(), UTIL::LCTOOLS::printTrackerPulse(), UTIL::LCTOOLS::printTrackerRawData(), UTIL::LCTOOLS::printTracks(), and UTIL::LCTOOLS::printVertices().

7.45.2.4 virtual bool EVENT::LCCollection::isDefault ( ) const [pure virtual]

True if collection is the default collection for the given type.

This implies that the collection is complete and unambigous. Convenient method that checks bit BITDefault of the flag word.

Implemented in **IMPL::LCCollectionVec** (p. 90).

7.45.2.5 virtual bool EVENT::LCCollection::isSubset ( ) const [pure virtual]

True if the collection holds a subset of objects from other collections.

If the collection is not transient only the pointers/references to the original objects will be stored. Convenient method that checks bit BITSubset of the flag word.

Implemented in **IMPL::LCCollectionVec** (p. 90).

7.45.2.6 virtual bool EVENT::LCCollection::isTransient ( ) const [pure virtual]

True if collection is transient, i.e.

will not be written to any **LCIO** (p. 115) file. Convenient method that checks bit BITTransient of the flag word.

Implemented in **IMPL::LCCollectionVec** (p. 90).

Referenced by SIO::SIOWriter::setUpHandlers().

7.45.2.7 virtual LCParameters& EVENT::LCCollection::parameters ( ) [pure virtual]

Parameters defined for this collection.

Can be used to modify the collection paramters.

Implemented in **IMPL::LCCollectionVec** (p. 88).

Referenced by UTIL::CellIDEncoder< T >::CellIDEncoder(), SIO::SIOVertexHandler::init(), SIO::SIOLCGenericObjectHandler::init(), SIO::SIOObjectHandler::init(), UTIL::CollectionParameterMap::∼CollectionParameterMap(), and UTIL::PIDHandler::∼PIDHandler().

7.45.2.8 virtual void EVENT::LCCollection::removeElementAt (int *i*) throw ReadOnlyException, std::exception) [pure virtual]

Removes the i-th element from the collection.

Throws an exception if the collection (event) is 'read only'.

Exceptions

ReadOnlyException *(p. 192)*	

Implemented in **IMPL::LCCollectionVec** (p. 90).

7.45.2.9 virtual void EVENT::LCCollection::setFlag (int *flag*) [pure virtual]

Set the flag word.

This is allowed in 'read only' mode.

Implemented in **IMPL::LCCollectionVec** (p. 88).

Referenced by SIO::SIOVertexHandler::init(), SIO::SIOLCGenericObjectHandler::init(), SIO::SIOObjectHandler::init(), and UTIL::CellIDEncoder< T >::setCellIDFlag().

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

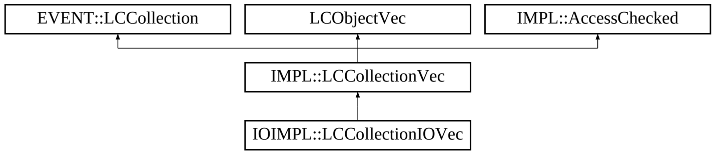
- LCCollection.h

7.46 IOIMPL::LCCollectionIOVec Class Reference

Adding stuff needed for io (friend declarations, etc.)

```
#include <IOIMPL/LCCollectionIOVec.h>
```

Inheritance diagram for IOIMPL::LCCollectionIOVec:



Public Member Functions

- **LCCollectionIOVec** (const std::string &type)

Friends

- class **SIO::SIOCollectionHandler**

Additional Inherited Members

7.46.1 Detailed Description

Adding stuff needed for io (friend declarations, etc.)

Author

gaede

Version

Mar 6, 2003

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

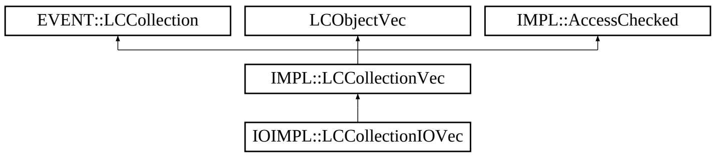
- LCCollectionIOVec.h

7.47 IMPL::LCCollectionVec Class Reference

Implementation of the LCCollection using (inheriting from) an STL vector of LCObjects.

`#include <IMPL/LCCollectionVec.h>`

Inheritance diagram for IMPL::LCCollectionVec:



Public Member Functions

- **LCCollectionVec** ()

  Default Constructor should be protected - every LCCollection needs to know the type of its elements.
- **LCCollectionVec** (const std::string &type)

  The public default constructur that takes the name of the type of the elements.
- virtual ∼**LCCollectionVec** ()

  Destructor.
- virtual int **getNumberOfElements** () const

  Returns the number of entries in the collection.
- virtual const std::string & **getTypeName** () const

  Returns the type name of the collection - valid names are defined in LCIO.
- virtual **EVENT::LCObject** ∗ **getElementAt** (int index) const

  Returns pointer to element at index - no range check !.
- virtual int **getFlag** () const

  Returns flag word for collection.
- virtual bool **isTransient** () const

  True if collection is transient, i.e.
- void **setTransient** (bool val=true)

  Sets the transient flag for this collection.
- virtual bool **isDefault** () const

  True if collection is the default collection for the given type.
- void **setDefault** (bool val=true)

  Sets the default flag for this collection.
- virtual bool **isSubset** () const

  True if the collection holds a subset of objects from other collections.
- void **setSubset** (bool val=true)

  Sets the subset flag for this collection.
- void **setFlag** (int flag)

  Sets the flag word for this collection.
- virtual void **addElement** (**EVENT::LCObject** ∗obj) throw (EVENT::ReadOnlyException)

  Adds the given element to (end of) the collection.
- virtual void **removeElementAt** (int i) throw (EVENT::ReadOnlyException)

  Removes the i-th element from the collection.
- virtual const **EVENT::LCParameters** & **getParameters** () const

  Parameters defined for this run.
- virtual **EVENT::LCParameters** & **parameters** ()

  Parameters defined for this run.

Protected Member Functions

- void **setReadOnly** (bool readOnly)

Protected Attributes

- std::string **_typeName**
- int **_flag**
- **LCParametersImpl _params**

Additional Inherited Members

7.47.1 Detailed Description

Implementation of the LCCollection using (inheriting from) an STL vector of LCObjects.

Author

gaede

Version

Id:

LCCollectionVec.h (p. **??**),v 1.12 2010-01-21 16:27:36 gaede Exp

See Also

LCObject
LCCollection

7.47.2 Constructor & Destructor Documentation

7.47.2.1 IMPL::LCCollectionVec::LCCollectionVec (const std::string & *type*)

The public default constructur that takes the name of the type of the elements.

7.47.3 Member Function Documentation

7.47.3.1 void IMPL::LCCollectionVec::addElement (EVENT::LCObject * *obj*) throw EVENT::ReadOnlyException) [virtual]

Adds the given element to (end of) the collection.

Throws an exception if the collection (event) is 'read only'.

Exceptions

ReadOnlyException	

Implements **EVENT::LCCollection** (p. 84).

Referenced by UTIL::LCRelationNavigator::createLCCollection().

7.47.3.2 int IMPL::LCCollectionVec::getFlag ( ) const [virtual]

Returns flag word for collection.

Bits 16-31 are reserved for LCIO Depending on the object type stored they have a special meaning, e.g. for SimCalorimeterHits:

CHBIT_LONG = 31 - store position

CHBIT_BARREL = 30 - endcap or barrel

CHBIT_ID1 = 29 - cellid1 is sored

CHBIT_PDG = 28 - store pdg of secondaries

Bit 16 is used to flag collection as transient

Bit 17 is used to flag collection as default

Bit 18 is used to flag collection as subset

Bits 0-15 are subdetector/user specific.

See Also

isTransient() (p. 90)

Implements **EVENT::LCCollection** (p. 84).

7.47.3.3 bool IMPL::LCCollectionVec::isDefault ( ) const [virtual]

True if collection is the default collection for the given type.

This implies that the collection is complete and unambigous. Convenient method that checks bit BITDefault of the flag word.

Implements **EVENT::LCCollection** (p. 85).

7.47.3.4 bool IMPL::LCCollectionVec::isSubset ( ) const [virtual]

True if the collection holds a subset of objects from other collections.

If the collection is not transient only the pointers/references to the original objects will be stored. Convenient method that checks bit BITSubset of the flag word.

Implements **EVENT::LCCollection** (p. 85).

Referenced by ∼LCCollectionVec().

7.47.3.5 bool IMPL::LCCollectionVec::isTransient ( ) const [virtual]

True if collection is transient, i.e.

will not be written to any LCIO file. Convenient method that checks bit 16 of the flag word.

Implements **EVENT::LCCollection** (p. 86).

References EVENT::LCCollection::BITTransient.

7.47.3.6 void IMPL::LCCollectionVec::removeElementAt (int *i*) throw EVENT::ReadOnlyException) [virtual]

Removes the i-th element from the collection.

Throws an exception if the collection (event) is 'read only'.

Exceptions

ReadOnlyException	

Implements **EVENT::LCCollection** (p. 86).

7.47.3.7 void IMPL::LCCollectionVec::setDefault (bool *val* = true)

Sets the default flag for this collection.

User need to make sure this flag is unique for a type.

7.47.3.8 void IMPL::LCCollectionVec::setSubset (bool *val* = `true`)

Sets the subset flag for this collection.

Collections of subsets are only written as pointers to LCIO files, i.e. the objects have to exist in another non-transient collections.

NB: Do not add any 'real' objects to a collection flagged as subset, as this will lead to memory leaks !

7.47.3.9 void IMPL::LCCollectionVec::setTransient (bool *val* = `true`)

Sets the transient flag for this collection.

Transient collections are not written to LCIO files.

References EVENT::LCCollection::BITTransient.

Referenced by IMPL::LCEventImpl::takeCollection().

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

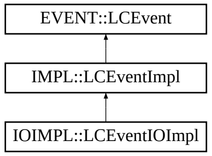
- LCCollectionVec.h
- LCCollectionVec.cc

7.48 EVENT::LCEvent Class Reference

The main event interface.

`#include <pre-generated/EVENT/LCEvent.h>`

Inheritance diagram for EVENT::LCEvent:



Public Member Functions

- virtual ∼**LCEvent** ()
  Destructor.
- virtual int **getRunNumber** () const =0
  Return the run number off this event.
- virtual int **getEventNumber** () const =0
  Returns this event's number .
- virtual const std::string & **getDetectorName** () const =0
  Returns the name of the detector setup used in the simulation.
- virtual **long64 getTimeStamp** () const =0
  Returns the time stamp of the event.
- virtual double **getWeight** () const =0
  Returns the event weight.
- virtual const std::vector < std::string > ∗ **getCollectionNames** () const =0

  Returns the names of the collections in the event.
- virtual **LCCollection** ∗ **getCollection** (const std::string &name) const =0 throw (DataNotAvailableException, std::exception)
  Returns the collection for the given name.
- virtual **LCCollection** ∗ **takeCollection** (const std::string &name) const =0 throw (DataNotAvailableException, std::exception)
  Returns the collection for the given name and transfers the ownership of the collection to the caller.
- virtual void **addCollection** (**LCCollection** ∗col, const std::string &name)=0 throw (EventException, std::exception)
  Adds a collection with the given name (has to be a valid C/C++ variable name).
- virtual void **removeCollection** (const std::string &name)=0 throw (ReadOnlyException, std::exception)
  Removes (and deletes) the collection with name (if it exists in the event).
- virtual const **LCParameters** & **getParameters** () const =0
  Parameters defined for this event.
- virtual **LCParameters** & **parameters** ()=0
  Parameters defined for this event.

7.48.1 Detailed Description

The main event interface.

Holds named collections with elements of type **LCObject** (p. 123) ('untyped'). Users have to cast to the corresponding types.

Author

gaede

Version

Id:

LCEvent.aid,v 1.14 2007-11-08 19:35:37 gaede Exp

See Also

LCCollection (p. 83)

7.48.2 Member Function Documentation

7.48.2.1 virtual void EVENT::LCEvent::addCollection (LCCollection ∗ *col*, const std::string & *name*) throw EventException, std::exception) `[pure virtual]`

Adds a collection with the given name (has to be a valid C/C++ variable name).

Throws an exception if the name already exists in the event. NB: Adding collections is allowed even when the event is 'read only'.

See Also

validateCollectionName

Exceptions

EventException *(p. 69)*	

Implemented in **IMPL::LCEventImpl** (p. 96).

7.48.2.2 virtual LCCollection∗ EVENT::LCEvent::getCollection (const std::string & *name*) const throw DataNotAvailableException, std::exception) [pure virtual]

Returns the collection for the given name.

Exceptions

DataNotAvailable-Exception *(p. 66)*	

Implemented in **IMPL::LCEventImpl** (p. 96).

Referenced by UTIL::LCTOOLS::dumpEvent(), UTIL::LCTOOLS::dumpEventDetailed(), and SIO::SIOWriter::setUpHandlers().

7.48.2.3 virtual long64 EVENT::LCEvent::getTimeStamp ( ) const [pure virtual]

Returns the time stamp of the event.

Timestamp is a unique 64 bit integer.

Definition: ns since January 1, 1970, 00:00:00 GMT

Implemented in **IMPL::LCEventImpl** (p. 94).

Referenced by UTIL::LCTOOLS::dumpEventDetailed().

7.48.2.4 virtual void EVENT::LCEvent::removeCollection (const std::string & *name*) throw ReadOnlyException, std::exception) [pure virtual]

Removes (and deletes) the collection with name (if it exists in the event).

Throws an exception if the event is 'read only' as defined by the read mode in LCReader.

Exceptions

ReadOnlyException *(p. 192)*	

Implemented in **IMPL::LCEventImpl** (p. 97).

7.48.2.5 virtual LCCollection∗ EVENT::LCEvent::takeCollection (const std::string & *name*) const throw DataNotAvailableException, std::exception) [pure virtual]

Returns the collection for the given name and transfers the ownership of the collection to the caller.

The caller is responsible for deleting the collection *after* the Event is deleted. The collection is still in the event but flagged as transient after the call returns.

This is usefull when you want to keep the collection for the next events.

Use with care!

Exceptions

DataNotAvailable-Exception *(p. 66)*	

Implemented in **IMPL::LCEventImpl** (p. 97).

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

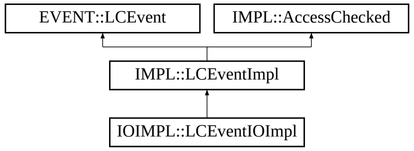
- LCEvent.h

7.49 IMPL::LCEventImpl Class Reference

Implementation of the main event class.

`#include <IMPL/LCEventImpl.h>`

Inheritance diagram for IMPL::LCEventImpl:



Public Member Functions

- virtual ∼**LCEventImpl** ()
  Copy contructor, creates a deep copy of the event.
- virtual int **getRunNumber** () const
  Return the run number off this event.
- virtual int **getEventNumber** () const
  Returns this event's number .
- virtual const std::string & **getDetectorName** () const
  Returns the name of the detector setup used in the simulation.
- virtual **EVENT::long64 getTimeStamp** () const
  Returns the time stamp of the event.
- virtual double **getWeight** () const
  Returns the event weight.
- virtual const std::vector < std::string > ∗ **getCollectionNames** () const
  Returns the names of the collections in the event.
- virtual **EVENT::LCCollection** ∗ **getCollection** (const std::string &name) const throw (EVENT::DataNotAvailableException, std::exception)
  Returns the collection for the given name.
- virtual **EVENT::LCCollection** ∗ **takeCollection** (const std::string &name) const throw (EVENT::DataNotAvailableException, std::exception)
  Returns the collection for the given name and transfers the ownership of the collection to the caller.
- virtual void **addCollection** (**EVENT::LCCollection** ∗col, const std::string &name) throw (EVENT::EventException, std::exception)
  Adds a collection with the given name (has to be a valid C/C++ variable name).

- virtual void **removeCollection** (const std::string &name) throw (EVENT::ReadOnlyException, std::exception)
  Removes (and deletes) the collection with name (if it exists in the event).
- virtual const **EVENT::LCParameters** & **getParameters** () const
  Returns the names of the relations in the event.
- virtual **EVENT::LCParameters** & **parameters** ()
  Parameters defined for this run.
- void **setRunNumber** (int rn)
  Sets the run number.
- void **setEventNumber** (int en)
  Sets the event number.
- void **setDetectorName** (const std::string &dn)
  Sets the detector name.
- void **setTimeStamp** (**EVENT::long64** ts)
  Sets the event time stamp.
- void **setWeight** (double w)
  Set the event weight.

Protected Member Functions

- void **setAccessMode** (int accessMode)
- bool **validateCollectionName** (const char *name)
  Tests the validity of a collection name.

Protected Attributes

- int **_runNumber**
- int **_eventNumber**
- **EVENT::long64 _timeStamp**
- std::string **_detectorName**
- LCCollectionMap **_colMap**
- std::vector< std::string > **_colNames**
- **LCParametersImpl _params**
- LCCollectionSet **_notOwned**

7.49.1 Detailed Description

Implementation of the main event class.

This is used by LCIO for reading events from file.

Author

gaede

Version

Mar 5, 2003

See Also

LCEvent
LCCollection

7.49.2 Constructor & Destructor Documentation

7.49.2.1 IMPL::LCEventImpl::∼LCEventImpl ( ) [virtual]

Copy contructor, creates a deep copy of the event.

Not yet - needs pointer chasing ...Destructor.

7.49.3 Member Function Documentation

7.49.3.1 void IMPL::LCEventImpl::addCollection (EVENT::LCCollection * *col*, const std::string & *name*) throw EVENT::EventException, std::exception) [virtual]

Adds a collection with the given name (has to be a valid C/C++ variable name).

Throws an exception if the name already exists in the event. NB: Adding collections is allowed even when the event is 'read only'.

See Also

validateCollectionName (p. 97)

Exceptions

EventException	

Implements **EVENT::LCEvent** (p. 92).

Referenced by UTIL::LCStdHepRdr::updateNextEvent(), and UTIL::LCStdHepRdrNew::updateNextEvent().

7.49.3.2 LCCollection * IMPL::LCEventImpl::getCollection (const std::string & *name*) const throw EVENT::DataNotAvailableException, std::exception) [virtual]

Returns the collection for the given name.

Exceptions

DataNotAvailableException	

Implements **EVENT::LCEvent** (p. 93).

Referenced by HEPEVTIMPL::HEPEVT::fromHepEvt().

7.49.3.3 virtual const EVENT::LCParameters& IMPL::LCEventImpl::getParameters ( ) const [inline], [virtual]

Returns the names of the relations in the event.

Returns the relation for the given name.

Exceptions

DataNotAvailableException	Adds a relation with the given name. Throws an exception if the name already exists in the event. NB: Adding relations is allowed even when the event is 'read only'.

EventException	Removes (and deletes) the relation with name (if it exists in the event). Throws an exception if the event is 'read only' as defined by the read mode in LCReader.
ReadOnlyException-Parameters	defined for this event.

Implements **EVENT::LCEvent** (p. 92).

7.49.3.4 void IMPL::LCEventImpl::removeCollection (const std::string & *name*) throw **EVENT::ReadOnlyException**, std::exception) `[virtual]`

Removes (and deletes) the collection with name (if it exists in the event).

Throws an exception if the event is 'read only' as defined by the read mode in LCReader.

Exceptions

ReadOnlyException	

Implements **EVENT::LCEvent** (p. 93).

7.49.3.5 **LCCollection** ∗ IMPL::LCEventImpl::takeCollection (const std::string & *name*) const throw **EVENT::DataNotAvailableException**, std::exception) `[virtual]`

Returns the collection for the given name and transfers the ownership of the collection to the caller.

The caller is responsible for deleting the collection *after* the Event is deleted. The collection is still in the event but flagged as transient after the call returns.

This is usefull when you want to keep the collection for the next events.

Use with care!

Exceptions

DataNotAvailableException	

Implements **EVENT::LCEvent** (p. 93).

References IMPL::LCCollectionVec::setTransient().

7.49.3.6 bool IMPL::LCEventImpl::validateCollectionName (const char ∗ *name*) `[protected]`

Tests the validity of a collection name.

LCIO only accepts collection names starting with (regular expression) [A-Za-z_] and continuing with `A-Za-z0-9-_`.

SIO (p. 30) only accepts names starting with (regular expression) [A-Za-z_] and continuing with `A-Za-z0-9_`.

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

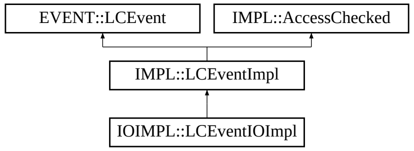
- LCEventImpl.h
- LCEventImpl.cc

7.50 IOIMPL::LCEventIOImpl Class Reference

Adding stuff needed for io (friend declarations, etc.)

`#include <IOIMPL/LCEventIOImpl.h>`

Inheritance diagram for IOIMPL::LCEventIOImpl:



Friends

- class **SIO::SIOReader**
- class **SIO::SIOEventHeaderHandler**
- class **SIO::SIOEventHandler**

Additional Inherited Members

7.50.1 Detailed Description

Adding stuff needed for io (friend declarations, etc.)

Author

gaede

Version

Mar 6, 2003

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

- LCEventIOImpl.h

7.51 IO::LCEventListener Class Reference

Listener for the occurence of LCEvents when reading a stream.

`#include <pre-generated/IO/LCEventListener.h>`

Public Member Functions

- virtual ∼**LCEventListener** ()

  Destructor.
- virtual void **processEvent** (**EVENT::LCEvent** ∗evt)=0

  Call back for analyzing an LCEvent.
- virtual void **modifyEvent** (**EVENT::LCEvent** ∗evt)=0

  Call back for updating an LCEvent.

7.51.1 Detailed Description

Listener for the occurence of LCEvents when reading a stream.

User classes should implement this interface for their analysis code and then register the listener with **LCReader** (p. 130).

Author

gaede

Version

Mar 11, 2003

See Also

LCEvent
LCReader (p. 130)

7.51.2 Member Function Documentation

7.51.2.1 virtual void IO::LCEventListener::modifyEvent (**EVENT::LCEvent** ∗ *evt*) [pure virtual]

Call back for updating an LCEvent.

Here all modifications are allowed. Only use this method if you need to remove a collection from the event or you need to modify (fix) exisiting data.

7.51.2.2 virtual void IO::LCEventListener::processEvent (**EVENT::LCEvent** ∗ *evt*) [pure virtual]

Call back for analyzing an LCEvent.

The access to the event is 'read only'. Adding new collections to the event is allowed wheras existing data collections and elements can not be modified.

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

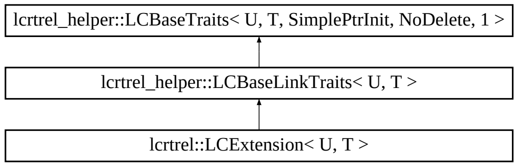
- LCEventListener.h

7.52 lcrtrel::LCExtension< U, T > Struct Template Reference

Simple Extension - pointer to an object of type T.

```
#include <LCRTRelations.h>
```

Inheritance diagram for lcrtrel::LCExtension< U, T >:



Additional Inherited Members

7.52.1 Detailed Description

template<class U, typename T>struct lcrtrel::LCExtension< U, T >

Simple Extension - pointer to an object of type T.

The class U needs to be the subclass type, e.g.

struct MyAttributes : public LCExtension<MyAttributes,Attributes> {} ;

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

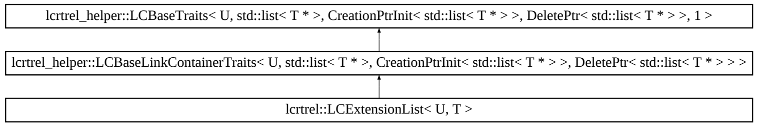
- LCRTRelations.h

7.53 lcrtrel::LCExtensionList< U, T > Class Template Reference

Extension list holding pointers to objects of type T - no ownership of the objects is taken.

```
#include <LCRTRelations.h>
```

Inheritance diagram for lcrtrel::LCExtensionList< U, T >:



Additional Inherited Members

7.53.1 Detailed Description

template<class U, class T>class lcrtrel::LCExtensionList< U, T >

Extension list holding pointers to objects of type T - no ownership of the objects is taken.

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

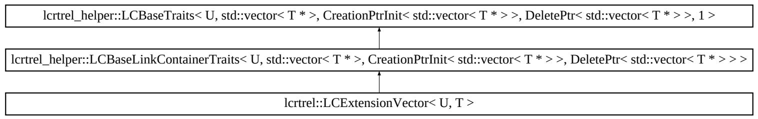
- LCRTRelations.h

7.54 lcrtrel::LCExtensionVector< U, T > Class Template Reference

Extension vector holding pointers to objects of type T - no ownership of the objects is taken.

```
#include <LCRTRelations.h>
```

Inheritance diagram for lcrtrel::LCExtensionVector< U, T >:



Additional Inherited Members

7.54.1 Detailed Description

template< class U, class T >class lcrtrel::LCExtensionVector< U, T >

Extension vector holding pointers to objects of type T - no ownership of the objects is taken.

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

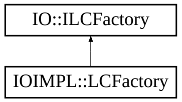
- LCRTRelations.h

7.55 IOIMPL::LCFactory Class Reference

Factory to create LCWriter and Reader object for the known data formats, e.g.

`#include <IOIMPL/LCFactory.h>`

Inheritance diagram for IOIMPL::LCFactory:



Public Member Functions

- virtual ∼**LCFactory** ()
  Destructor.
- virtual **IO::LCWriter** ∗ **createLCWriter** ()
  Creates an LCWriter object for the current persistency type.
- virtual **IO::LCReader** ∗ **createLCReader** (int lcReaderFlag=0)
  Creates an LCReader object for the current persistency type.

Static Public Member Functions

- static **LCFactory** ∗ **getInstance** ()
  Returns the instance of the factory.

7.55.1 Detailed Description

Factory to create LCWriter and Reader object for the known data formats, e.g.

SIO (p. 30). Singleton class that knows all concrete reader/writer implementations.

Author

gaede

Version

Mar 10, 2003

7.55.2 Member Function Documentation

7.55.2.1 LCReader ∗ IOIMPL::LCFactory::createLCReader (int *lcReaderFlag* = 0) [virtual]

Creates an LCReader object for the current persistency type.

lcReaderFlag: configuration options for the LCReader object - combine multible options with '|'. So far only LC-Reader::directAccess.

Implements **IO::ILCFactory** (p. 72).

7.55.2.2 LCFactory ∗ IOIMPL::LCFactory::getInstance ( ) [static]

Returns the instance of the factory.

In the future we need to specify the data format here... This doesn't work for java as an interface can't have static members ...

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

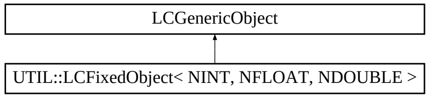
- LCFactory.h
- LCFactory.cc

7.56 UTIL::LCFixedObject< NINT, NFLOAT, NDOUBLE > Class Template Reference

Template class for fixed size LCGenericObject subclasses.

`#include <UTIL/LCFixedObject.h>`

Inheritance diagram for UTIL::LCFixedObject< NINT, NFLOAT, NDOUBLE >:



Public Member Functions

- **LCFixedObject** ()
  Default c'tor.
- **LCFixedObject** (**LCObject** ∗**obj**)
  C'tor to be used for elements of LCGenericObjects read from an LCIO file or the database.
- LCGenericObjectImpl ∗ **obj** ()
  The LCGenericObjectImpl .
- virtual ∼**LCFixedObject** ()
  Clean up if we created a new LCGenericObjectImpl.
- virtual int **id** () const
  Return the id of the underlying LCGenericObjectImpl.
- int **getNInt** () const
- int **getNFloat** () const
- int **getNDouble** () const
- int **getIntVal** (int index) const
- float **getFloatVal** (int index) const
- double **getDoubleVal** (int index) const
- bool **isFixedSize** () const

Protected Attributes

- LCGenericObjectImpl ∗ **_obj**
- bool **_createdObject**

7.56.1 Detailed Description

template<int NINT, int NFLOAT, int NDOUBLE>class UTIL::LCFixedObject< NINT, NFLOAT, NDOUBLE >

Template class for fixed size LCGenericObject subclasses.

By inheriting from this class users can very easily implement their own objects.

It uses an instance of LCGenericObjectImpl that holds the data, thus there is no overhead when the data is read from a database or file for copying it to some local structure (Decorator pattern).

This is still an abstract class: subclasses have to implement LCGenericObject::getTypeName() and LCGeneric-Object::getDataDescription().

Author

gaede

Version

Id:

LCFixedObject.h (p. **??**),v 1.3 2011-03-13 12:41:49 gaede Exp

7.56.2 Constructor & Destructor Documentation

7.56.2.1 template<int NINT, int NFLOAT, int NDOUBLE> UTIL::LCFixedObject< NINT, NFLOAT, NDOUBLE >::LCFixedObject (LCObject ∗ *obj*) `[inline]`

C'tor to be used for elements of LCGenericObjects read from an LCIO file or the database.

Subclasses should 'override' this, e.g.:

Myclass(LCObject∗ obj) : LCFixedObject(obj) {}

References UTIL::LCFixedObject< NINT, NFLOAT, NDOUBLE >::obj().

7.56.3 Member Function Documentation

7.56.3.1 template<int NINT, int NFLOAT, int NDOUBLE> LCGenericObjectImpl∗ UTIL::LCFixedObject< NINT, NFLOAT, NDOUBLE >::obj ( ) `[inline]`

The LCGenericObjectImpl .

Sublcasses use this to access their data.

Referenced by UTIL::LCFixedObject< NINT, NFLOAT, NDOUBLE >::LCFixedObject().

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

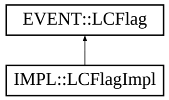
- LCFixedObject.h

7.57 EVENT::LCFlag Class Reference

Helper class to create and interpret the 32-bit flag word in LCCollections.

`#include <pre-generated/EVENT/LCFlag.h>`

Inheritance diagram for EVENT::LCFlag:



Public Member Functions

- virtual ∼**LCFlag** ()
  Destructor.
- virtual bool **bitSet** (int index) const =0
  Returns true if bit at given index is set.
- virtual int **getFlag** () const =0
  Returns the flag word.
- virtual void **setBit** (int bit)=0
  Sets bit to 1.
- virtual void **unsetBit** (int bit)=0
  Sets bit to 0.

7.57.1 Detailed Description

Helper class to create and interpret the 32-bit flag word in LCCollections.

Bits are defined in class **LCIO** (p. 115).

See Also

LCIO (p. 115)

Author

gaede

Version

May 6, 2003

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

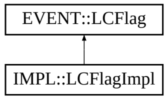
- LCFlag.h

7.58 IMPL::LCFlagImpl Class Reference

Implementation of helper class to create and interpret the 32-bit flag word in LCCollections.

`#include <IMPL/LCFlagImpl.h>`

Inheritance diagram for IMPL::LCFlagImpl:



Public Member Functions

- **LCFlagImpl** ()
  Constructor initializing flag with 0.
- **LCFlagImpl** (int flag)
  Constructor for evaluating an existing flag word.
- virtual ~**LCFlagImpl** ()
  Destructor.
- virtual bool **bitSet** (int index) const
  Returns true if bit at given index is set.
- virtual int **getFlag** () const
  Returns the flag word.
- virtual void **setBit** (int bit)
  Sets bit to 1.
- virtual void **unsetBit** (int bit)
  Sets bit to 0.

7.58.1 Detailed Description

Implementation of helper class to create and interpret the 32-bit flag word in LCCollections.

Bits are defined in class LCIO.

See Also

LCIO

Author

gaede

Version

May 6, 2003

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

- LCFlagImpl.h
- LCFlagImpl.cc

7.59 lcrtrel::LCFloatExtension< U > Struct Template Reference

Public Types

- typedef double **ptr**
- typedef double & **ext_type**
- typedef U **tag**

Static Public Member Functions

- static void **clean** (void ∗)
- static ptr **init** ()
- static DeleteFPtr **deletePtr** ()

Static Public Attributes

- static const int **allowed_to_call_ext** = 1

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

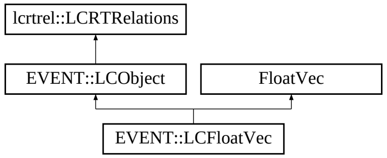
- LCRTRelations.h

7.60 EVENT::LCFloatVec Class Reference

Float vector used for user extensions.

`#include <pre-generated/EVENT/LCFloatVec.h>`

Inheritance diagram for EVENT::LCFloatVec:



Public Types

- typedef **LCFloatVec lcobject_type**

Public Member Functions

- virtual ~**LCFloatVec** ()
  Destructor.
- int **id** () const
  *Returns an object id for internal (debugging) use in **LCIO** (p. 115).*

Protected Attributes

- **IMPL::AccessChecked _acc**

Additional Inherited Members

7.60.1 Detailed Description

Float vector used for user extensions.

The cpp implementation is a std::vector<float>

Author

gaede

Version

Mar 12, 2003

See Also

LCObject (p. 123)

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

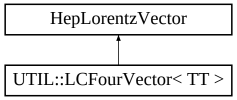
- pre-generated/EVENT/LCFloatVec.h

7.61 UTIL::LCFourVector< TT > Class Template Reference

Four vector used in LCIO.

`#include <UTIL/LCFourVector.h>`

Inheritance diagram for UTIL::LCFourVector< TT >:



Public Member Functions

- **LCFourVector** (const TT ∗**lcObj**)
  Constructor for templated type,e.g.
- **LCFourVector** (const **EVENT::LCObject** ∗**lcObj**)
  Constructor for LCObject.
- const TT ∗ **operator->** () const
  *Instances of **LCFourVector** (p. 107) serve as a handle to the templated type.*
- const TT ∗ **lcObj** () const
  Pointer to the LCObject that has been used to create the four vector.

Protected Attributes

- const TT ∗ **_lcObj**

7.61.1 Detailed Description

template<class TT>class UTIL::LCFourVector< TT >

Four vector used in LCIO.

The cpp implementation is a CLHEP::HepLorentzVector. Instances of this class can be used as CLHEP::HepLorentzVector AND pointers to the corresponding LCIO class.

Currently available are:

- LCFourVector<MCParticle>/MCParticle4V
- LCFourVector<ReconstructedParticle>/ReconstructedParticle4V

Author

gaede

Version

Mar 12, 2004

7.61.2 Constructor & Destructor Documentation

7.61.2.1 template<class TT> UTIL::LCFourVector< TT >::LCFourVector (const TT ∗ *lcObj*) `[inline]`

Constructor for templated type,e.g.

LCFourVector<MCParticle>(myMCParticle).

7.61.2.2 template<class TT> UTIL::LCFourVector< TT >::LCFourVector (const EVENT::LCObject ∗ *lcObj*) `[inline]`

Constructor for LCObject.

Can be used to save the dynamic_cast to the explicit type from an LCCollection element, e.g. LCFourVector<MCParticle>(col->getElement(i)).

7.61.3 Member Function Documentation

7.61.3.1 template<class TT> const TT∗ UTIL::LCFourVector< TT >::lcObj ( ) const `[inline]`

Pointer to the LCObject that has been used to create the four vector.

To be used when the object is needed as a function argument or optionally to call methods of the object, e.g. particle4V.lcObj()->getMomentum() ;

7.61.3.2 template<class TT> const TT∗ UTIL::LCFourVector< TT >::operator-> ( ) const `[inline]`

Instances of **LCFourVector** (p. 107) serve as a handle to the templated type.

Methods of the original type, e.g. MCParticle can be accessed via the '->' operator. Methods of the four vector implementation (HepLorentzvector) are accessed via the '.' operator.

For Example:

LCFourVector<MCParticle> particle4V(col->getElementAt(i))

particle4V->getMomentum() // LCIO object

particle4V.m() // mass from HepLorentzVector

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

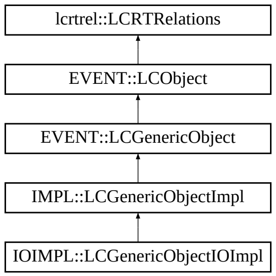
- LCFourVector.h

7.62 EVENT::LCGenericObject Class Reference

Simple interface to store generic user data.

`#include <pre-generated/EVENT/LCGenericObject.h>`

Inheritance diagram for EVENT::LCGenericObject:



Public Types

- typedef **LCGenericObject lcobject_type**

  Useful typedef for template programming with ***LCIO*** *(p. 115).*

Public Member Functions

- virtual ∼**LCGenericObject** ()

  Destructor.
- virtual int **getNInt** () const =0

  Number of integer values stored in this object.
- virtual int **getNFloat** () const =0

  Number of float values stored in this object.
- virtual int **getNDouble** () const =0

  Number of double values stored in this object.
- virtual int **getIntVal** (int index) const =0

  Returns the integer value for the given index.
- virtual float **getFloatVal** (int index) const =0

  Returns the float value for the given index.
- virtual double **getDoubleVal** (int index) const =0

  Returns the double value for the given index.
- virtual bool **isFixedSize** () const =0

  True if objects of the implementation class have a fixed size, i.e getNInt, getNFloat and getNDouble will return values that are constant during the lifetime of the object.
- virtual const std::string **getTypeName** () const =0

  The type name of the user class (typically the class name) This type name is stored as a collection parameter "TypeName" with every collection of ***LCGenericObject*** *(p. 109) subclasses.*
- virtual const std::string **getDataDescription** () const =0

  The description string.
- virtual int **id** () const

  Provide default implementation for id.

Additional Inherited Members

7.62.1 Detailed Description

Simple interface to store generic user data.

To store your own classes they have to implement this interface.

2005-02-11, gaede: Changed the return value of **getTypeName()** (p. 110) and **getDataDescription()** (p. 110) from const std::string& to const std::string to make the implementation easier

Author

gaede

Version

Id:

LCGenericObject.aid,v 1.7 2006-08-04 16:52:44 gaede Exp

7.62.2 Member Function Documentation

7.62.2.1 virtual const std::string EVENT::LCGenericObject::getDataDescription ( ) const `[pure virtual]`

The description string.

A comma separated list of pairs of type identifier, one of 'i','f','d' followed by ':' and an attribute name, e.g. "i:cellId,f-:offset,f:gain".

Implemented in **IMPL::LCGenericObjectImpl** (p. 112).

7.62.2.2 virtual const std::string EVENT::LCGenericObject::getTypeName ( ) const `[pure virtual]`

The type name of the user class (typically the class name) This type name is stored as a collection parameter "TypeName" with every collection of **LCGenericObject** (p. 109) subclasses.

Implemented in **IMPL::LCGenericObjectImpl** (p. 111).

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

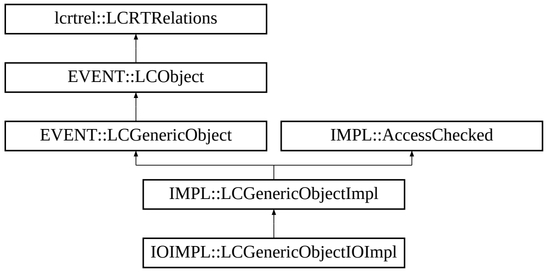
- LCGenericObject.h

7.63 IMPL::LCGenericObjectImpl Class Reference

Default LCIO implementation of the interface to store generic user data.

`#include <IMPL/LCGenericObjectImpl.h>`

Inheritance diagram for IMPL::LCGenericObjectImpl:



Public Member Functions

- **LCGenericObjectImpl** ()

  Variable size c'tor.
- **LCGenericObjectImpl** (int nInt, int nFloat, int nDouble)

  Fixed size c'tor.
- virtual ∼**LCGenericObjectImpl** ()

  Destructor.
- virtual int **id** () const

  Provide default implementation for id.
- virtual int **getNInt** () const

  Number of integer values stored in this object.
- virtual int **getNFloat** () const

  Number of float values stored in this object.
- virtual int **getNDouble** () const

  Number of double values stored in this object.
- virtual int **getIntVal** (int index) const

  Returns the integer value for the given index.
- virtual float **getFloatVal** (int index) const

  Returns the float value for the given index.
- virtual double **getDoubleVal** (int index) const

  Returns the double value for the given index.
- virtual void **setIntVal** (unsigned index, int value)

  Sets the integer value at the given index.
- virtual void **setFloatVal** (unsigned index, float value)

  Sets the float value at the given index.
- virtual void **setDoubleVal** (unsigned index, double value)

  Sets the double value at the given index.
- virtual bool **isFixedSize** () const

  True if objects of the implementation class have a fixed size, i.e getNInt, getNFloat and getNDouble will return values that are constant during the lifetime of the object.
- virtual const std::string **getTypeName** () const

  The type name of the user class (typically the class name)
- virtual const std::string **getDataDescription** () const

  The description string.

Protected Attributes

- std::vector< int > **_intVec**
- std::vector< float > **_floatVec**
- std::vector< double > **_doubleVec**
- bool **_isFixedSize**

Static Protected Attributes

- static std::string **_typeName** = "LCGenericObject"
- static std::string **_dataDescription** = ""

Additional Inherited Members

7.63.1 Detailed Description

Default LCIO implementation of the interface to store generic user data.

Author

gaede

Version

Id:

LCGenericObjectImpl.h (p. **??**),v 1.5 2006-08-04 16:52:46 gaede Exp

7.63.2 Member Function Documentation

7.63.2.1 const std::string IMPL::LCGenericObjectImpl::getDataDescription ( ) const [virtual]

The description string.

A comma separated list of pairs of type identifier, one of 'i','f','d' followed by ':' and an attribute name, e.g. "i:cellId,f-:offset,f:gain".

Implements **EVENT::LCGenericObject** (p. 110).

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

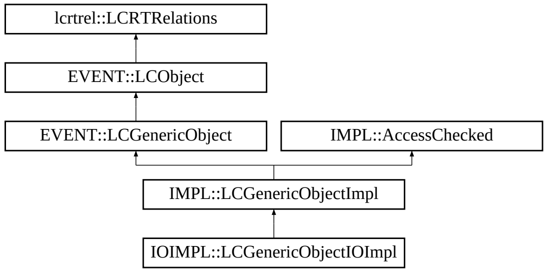
- LCGenericObjectImpl.h
- LCGenericObjectImpl.cc

7.64 IOIMPL::LCGenericObjectIOImpl Class Reference

Adding stuff needed for io (friend declarations, etc.)

`#include <IOIMPL/LCGenericObjectIOImpl.h>`

Inheritance diagram for IOIMPL::LCGenericObjectIOImpl:



Public Member Functions

- **LCGenericObjectIOImpl** (int nInt, int nFloat, int nDouble)

Friends

- class **SIO::SIOLCGenericObjectHandler**

Additional Inherited Members

7.64.1 Detailed Description

Adding stuff needed for io (friend declarations, etc.)

Author

gaede

Version

Id:

LCGenericObjectIOImpl.h (p. **??**),v 1.2 2005-04-15 08:37:40 gaede Exp

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

- LCGenericObjectIOImpl.h

7.65 lcrtrel::LCIntExtension< U > Struct Template Reference

Special Extension that allows to write int extensions directly (not through a pointer !).

`#include <LCRTRelations.h>`

Public Types

- typedef long **ptr**
- typedef long & **ext_type**
- typedef U **tag**

Static Public Member Functions

- static void **clean** (void ∗)
- static ptr **init** ()
- static DeleteFPtr **deletePtr** ()

Static Public Attributes

- static const int **allowed_to_call_ext** = 1

7.65.1 Detailed Description

template< class U > struct lcrtrel::LCIntExtension< U >

Special Extension that allows to write int extensions directly (not through a pointer !).

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

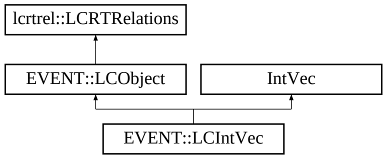
- LCRTRelations.h

7.66 EVENT::LCIntVec Class Reference

Int vector used for user extensions.

`#include <pre-generated/EVENT/LCIntVec.h>`

Inheritance diagram for EVENT::LCIntVec:



Public Types

- typedef **LCIntVec lcobject_type**

Public Member Functions

- virtual ∼**LCIntVec** ()

  Destructor.
- int **id** () const

  *Returns an object id for internal (debugging) use in **LCIO** (p. 115).*

Protected Attributes

- **IMPL::AccessChecked _acc**

Additional Inherited Members

7.66.1 Detailed Description

Int vector used for user extensions.

The cpp implementation is a std::vector<int>

Author

gaede

Version

Aug 8, 2003

See Also

LCObject (p. 123)

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

- pre-generated/EVENT/LCIntVec.h

7.67 EVENT::LCIO Class Reference

Global constants used in **LCIO** (p. 115).

```
#include <pre-generated/EVENT/LCIO.h>
```

Public Member Functions

- virtual ~**LCIO** ()
  Destructor.

Static Public Attributes

- static const int **MAJORVERSION** = LCIO_MAJOR_VERSION
- static const int **MINORVERSION** = LCIO_MINOR_VERSION
- static const int **CHBIT_LONG** = 31
- static const int **CHBIT_BARREL** = 30
- static const int **CHBIT_ID1** = 29
- static const int **CHBIT_PDG** = 28
- static const int **CHBIT_STEP** = 28
- static const int **RCHBIT_LONG** = 31
- static const int **RCHBIT_BARREL** = 30
- static const int **RCHBIT_ID1** = 29
- static const int **RCHBIT_NO_PTR** = 28
- static const int **RCHBIT_TIME** = 27
- static const int **RCHBIT_ENERGY_ERROR** = 26
- static const int **TRAWBIT_ID1** = 31
- static const int **TRAWBIT_CM** = 30
- static const int **RTHBIT_ID1** = 31
- static const int **RTHPBIT_ID1** = 31

- static const int **RTHZBIT_ID1** = 31
- static const int **THBIT_BARREL** = 31
- static const int **THBIT_MOMENTUM** = 30
- static const int **THBIT_ID1** = 29
- static const int **TRBIT_HITS** = 31
- static const int **CLBIT_HITS** = 31
- static const int **TPCBIT_RAW** = 31
- static const int **TPCBIT_NO_PTR** = 30
- static const int **LCREL_WEIGHTED** = 31
- static const int **GOBIT_FIXED** = 31
- static const int **READ_ONLY** = 0
- static const int **UPDATE** = 1
- static const int **ERROR** = 0
- static const int **SUCCESS** = 1
- static const int **WRITE_NEW** = 0
- static const int **WRITE_APPEND** = 1
- static const char * **LCEVENT** = "LCEvent"
- static const char * **LCRUNHEADER** = "LCRunHeader"
- static const char * **LCCOLLECTION** = "LCCollection"
- static const char * **MCPARTICLE** = "MCParticle"
- static const char * **SIMCALORIMETERHIT** = "SimCalorimeterHit"
- static const char * **RAWCALORIMETERHIT** = "RawCalorimeterHit"
- static const char * **CALORIMETERHIT** = "CalorimeterHit"
- static const char * **SIMTRACKERHIT** = "SimTrackerHit"
- static const char * **TPCHIT** = "TPCHit"
- static const char * **TRACKERRAWDATA** = "TrackerRawData"
- static const char * **TRACKERDATA** = "TrackerData"
- static const char * **TRACKERPULSE** = "TrackerPulse"
- static const char * **TRACKERHIT** = "TrackerHit"
- static const char * **TRACKERHITPLANE** = "TrackerHitPlane"
- static const char * **TRACKERHITZCYLINDER** = "TrackerHitZCylinder"
- static const char * **LCSTRVEC** = "LCStrVec"
- static const char * **LCFLOATVEC** = "LCFloatVec"
- static const char * **LCINTVEC** = "LCIntVec"
- static const char * **TRACK** = "Track"
- static const char * **TRACKSTATE** = "TrackState"
- static const char * **CLUSTER** = "Cluster"
- static const char * **RECONSTRUCTEDPARTICLE** = "ReconstructedParticle"
- static const char * **LCRELATION** = "LCRelation"
- static const char * **LCGENERICOBJECT** = "LCGenericObject"
- static const char * **PARTICLEID** = "ParticleID"
- static const char * **VERTEX** = "Vertex"
- static const char * **CellIDEncoding** = "CellIDEncoding"

7.67.1 Detailed Description

Global constants used in **LCIO** (p. 115).

Author

gaede

Version

Id:

LCIO.aid,v 1.77 2010-10-29 10:26:09 gaede Exp

See Also

LCObject (p. 123)
LCIO (p. 115)

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

- LCIO.h
- LCIO.cc

7.68 UTIL::LCIO_LONG< T > Class Template Reference

Public Member Functions

- **LCIO_LONG** (const T &o, const **EVENT::LCCollection** ∗c)
- const T ∗ **object** (void) const
- const **EVENT::LCCollection** ∗ **collection** (void) const

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

- Operators.h

7.69 UTIL::lcio_short< T > Class Template Reference

Public Member Functions

- **lcio_short** (const T ∗t, const **EVENT::LCCollection** ∗b=NULL)

Public Attributes

- const T ∗ **obj**
- const **EVENT::LCCollection** ∗ **col**

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

- Operators.h

7.70 IMPL::LCIOExceptionHandler Class Reference

Sets the default error handling for unexpected and uncaught exceptions.

`#include <IMPL/LCIOExceptionHandler.h>`

Static Public Member Functions

- static **LCIOExceptionHandler** ∗ **createInstance** ()

7.70.1 Detailed Description

Sets the default error handling for unexpected and uncaught exceptions.

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

- LCIOExceptionHandler.h
- LCIOExceptionHandler.cc

7.71 SIO::LCIORandomAccess Class Reference

Implementation class for **LCIORandomAccess** (p. 118) records.

`#include <SIO/LCIORandomAccess.h>`

Public Member Functions

- long64 **getIndexLocation** () const
- long64 **getPrevLocation** () const
- long64 **getFirstRecordLocation** () const
- void **setIndexLocation** (long64 il)
- void **setPreviousLocation** (long64 pl)
- void **setFirstRecordLocation** (long64 fl)

Protected Attributes

- **RunEvent _minRunEvt**
- **RunEvent _maxRunEvt**
- int **_nRunHeaders**
- int **_nEvents**
- int **_recordsAreInOrder**
- long64 **_indexLocation**
- long64 **_prevLocation**
- long64 **_nextLocation**
- long64 **_firstRecordLocation**

Friends

- class **SIORandomAccessHandler**
- class **LCIORandomAccessMgr**
- std::ostream & **operator<<** (std::ostream &os, const **LCIORandomAccess** &ra)
- bool **operator<** (const **LCIORandomAccess** ra0, const **LCIORandomAccess** &other)

7.71.1 Detailed Description

Implementation class for **LCIORandomAccess** (p. 118) records.

Author

gaede

Version

Id:

LCIORandomAccess.h (p. **??**),v 1.2 2010-06-22 13:49:54 gaede Exp

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

- LCIORandomAccess.h

7.72 SIO::LCIORandomAccessMgr Class Reference

Manager class for LCIO direct access.

```
#include <SIO/LCIORandomAccessMgr.h>
```

Public Member Functions

- long64 **getPosition** (const **RunEvent** &re)
  Return the position of the specified Event record or Run record respectively (if EventNum == -1).
- void **add** (const **RunEvent** &re, long64 pos)
  *Add a new entry to the event map - if the **RunEvent** (p. 202) already exists the new position will be stored.*
- bool **createEventMap** (SIO_stream *s)
  Get the run and event header map from the stream - either by reading the random access records or by recreating it for old files.
- const **RunEventMap** & **getEventMap** ()
  Return the event map - it will be empty, if not yet created.
- bool **initAppend** (SIO_stream *s)
  *Initialize random access for append mode: read last **LCIORandomAccess** (p. 118) record if it exists - recreate the **RunEvent** (p. 202) map from the file if not (old files).*
- void **writeRandomAccessRecords** (SIO_stream *stream)
  *Write the current random access records LCIOIndex and **LCIORandomAccess** (p. 118) to the stream.*
- void **clear** ()
  Clear all lists and maps before closing a file.

Protected Member Functions

- **LCIORandomAccess** * **createFromEventMap** ()
  *Prepare an **LCIORandomAccess** (p. 118) object from the current contents of **RunEventMap** (p. 202) (all file locations set to 0).*
- void **addLCIORandomAccess** (**LCIORandomAccess** *ra)
  *Add a new **LCIORandomAccess** (p. 118) object to the list.*
- bool **readLCIORandomAccessAt** (SIO_stream *stream, long64 pos)
  *Read the **LCIORandomAccess** (p. 118) record at the specified position.*
- bool **readLCIOIndexAt** (SIO_stream *stream, long64 pos)
  Read the LCIOIndex record at the specified position.
- bool **recreateEventMap** (SIO_stream *)
  *Fill the **RunEventMap** (p. 202) from the event and run header records in the file.*
- bool **readLCIORandomAccess** (SIO_stream *stream)
  *Helper for reading the next **LCIORandomAccess** (p. 118) record (need preceeding call to LCSIO::seek())*
- bool **readLCIOIndex** (SIO_stream *stream)
  Helper for reading the next LCIOIndex record (need preceeding call to LCSIO::seek())
- void **createFileRecord** ()
  *Create file record from all **LCIORandomAccess** (p. 118) records.*
- const **LCIORandomAccess** * **lastLCIORandomAccess** ()
  *Pointer to the last **LCIORandomAccess** (p. 118) in the list.*

Protected Attributes

- **RunEventMap _runEvtMap**
- std::list< **LCIORandomAccess** * > **_list**
- **LCIORandomAccess** * **_fileRecord**

Friends

- class **SIOIndexHandler**
- class **SIORandomAccessHandler**
- std::ostream & **operator**<< (std::ostream &os, const **LCIORandomAccessMgr** &ra)

7.72.1 Detailed Description

Manager class for LCIO direct access.

Provides the functionality for reading and writing the **LCIORandomAccess** (p. 118) and LCIOIndex records as needed by **SIOReader** (p. 234) and **SIOWriter** (p. 253). Direct Access is implemented through appending LCIOIndex and **LCIORandomAccess** (p. 118) and records to the end of the LCIO file. The LCIOIndex records hold the locations of the RunHeader and EventHeader in the file and are referenced by corresponding **LCIORandomAccess** (p. 118) records. The last record in the file will always be an **LCIORandomAccess** (p. 118) record - and previous records can be found through pointers to the previous record (simply linked list). **LCIORandomAccess** (p. 118) are always stored uncompressed. When reading a file with direct access mode a **RunEvent** (p. 202) map is created by reading all **LCIORandomAccess** (p. 118) and LCIOIndex records or - for old files - is recreated from all RunHeader and EventHeader records in the file. If an old file is opened in APPEND mode the corresponding records are created and written at the end on close()

See Also

writeRandomAccessRecords() (p. 119). : currently the last **LCIORandomAccess** (p. 118) record is found by seeking a fixed size from the end - need to store the actual length as last word in the file for future changes of the records length

Author

gaede

Version

Id:

LCIORandomAccessMgr.h (p. **??**),v 1.3 2011-03-03 16:00:12 gaede Exp

7.72.2 Member Function Documentation

7.72.2.1 **LCIORandomAccess** ∗ SIO::LCIORandomAccessMgr::createFromEventMap ( ) [protected]

Prepare an **LCIORandomAccess** (p. 118) object from the current contents of **RunEventMap** (p. 202) (all file locations set to 0).

Referenced by writeRandomAccessRecords().

7.72.2.2 long64 SIO::LCIORandomAccessMgr::getPosition (const **RunEvent** & *re*) [inline]

Return the position of the specified Event record or Run record respectively (if EventNum == -1).

Returns RunEventMap::NPos if no record found.

References SIO::RunEventMap::getPosition().

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

- LCIORandomAccessMgr.h
- LCIORandomAccessMgr.cc

7.73 UTIL::LCIterator< T > Class Template Reference

Simple convenient iterator class for LCCollections that saves some boiler plate code.

`#include <UTIL/LCIterator.h>`

Public Member Functions

- **LCIterator** (**EVENT::LCEvent** ∗evt, const std::string &name)
  Constructor that reads the collection with the given name from the event - note: no exception is thrown if the collection is not in the event, this will behave the same as an empty collection - use operator() to test, if the collection exists.
- **LCIterator** (**EVENT::LCCollection** ∗col)
  Constructor for the given collection.
- T ∗ **next** ()
  Returns the next element as long as there is one, otherwise 0 is returned.
- int **size** ()
  Size of the collection.
- **EVENT::LCCollection** ∗ **operator->** ()
  Serves as a handle to the LCCollection itself, to provide access to the collection parameters etc.
- **EVENT::LCCollection** ∗ **operator()** ()
  Return pointer to LCCollection, e.g.

7.73.1 Detailed Description

template<class T>class UTIL::LCIterator< T >

Simple convenient iterator class for LCCollections that saves some boiler plate code.

LCIterators can be constructed either form an LCCollection or from the LCEvent and and a collection name.

Examples:

```
// ------ use with while():
LCIterator<TrackerHit> it( evt, "VXDCollection" ) ;
while( TrackerHit* hit = it.next()  ){
```



```
   hitMap[ hit->getCellID0() ] = hit ;
 }

 // --- with for() :
 for(  LCIterator<Track> it( col ) ;  Track* trk = it.next()  ; ) {

   std::cout << trk->getTrackState( TrackState::AtIP ) << std::endl  ;
 }
```

Author

F.Gaede, DESY

Version

$Id:$

7.73.2 Member Function Documentation

7.73.2.1 template<class T> **EVENT::LCCollection**∗ UTIL::LCIterator< T >::operator() ( ) [inline]

Return pointer to LCCollection, e.g.

for testing whether the collections was in the event.

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

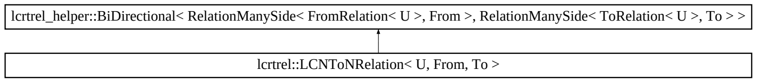
- LCIterator.h

7.74 lcrtrel::LCNToNRelation< U, From, To > Struct Template Reference

Many to many relation between objects of type From to objects of type To.

`#include <LCRTRelations.h>`

Inheritance diagram for lcrtrel::LCNToNRelation< U, From, To >:



Additional Inherited Members

7.74.1 Detailed Description

template<class U, class From, class To>struct lcrtrel::LCNToNRelation< U, From, To >

Many to many relation between objects of type From to objects of type To.

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

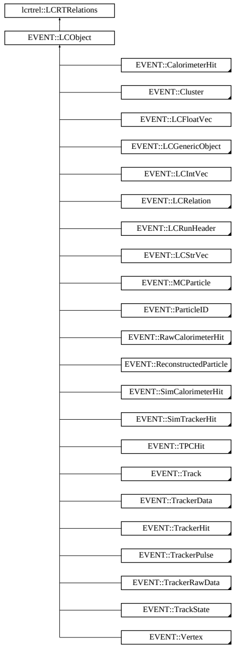
- LCRTRelations.h

7.75 EVENT::LCObject Class Reference

The generic object that is held in an **LCCollection** (p. 83).

`#include <pre-generated/EVENT/LCObject.h>`

Inheritance diagram for EVENT::LCObject:



Public Member Functions

- virtual ∼**LCObject** ()

  Destructor.
- virtual int **id** () const =0

  *Returns an object id for internal (debugging) use in **LCIO** (p. 115).*
- virtual **LCObject** ∗ **clone** () const

  *Returns an object id for internal (debugging) use in **LCIO** (p. 115).*

Additional Inherited Members

7.75.1 Detailed Description

The generic object that is held in an **LCCollection** (p. 83).

Every object stored in an **LCCollection** (p. 83), i.e. in an **LCEvent** (p. 91) has to implement this interface.

Author

gaede

Version

Jun 7, 2003 fg 20040405 moved here from hep.lcio.data

7.75.2 Member Function Documentation

7.75.2.1 virtual LCObject∗ EVENT::LCObject::clone ( ) const [inline], [virtual]

Returns an object id for internal (debugging) use in **LCIO** (p. 115).

Deep copy of the object. C++ implementations should return the concrete type (virtual constructor). No reasonable implementation yet. Do not use this method.

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

- LCObject.h

7.76 UTIL::LCObjectHandle< T > Class Template Reference

Template handle class for LCObjects.

`#include <UTIL/LCObjectHandle.h>`

Public Member Functions

- **LCObjectHandle** (T ∗lcObj)
- **LCObjectHandle** (**EVENT::LCObject** ∗lcObj)
- T ∗ **operator->** ()
- T ∗ **lcObj** ()

Protected Attributes

- T ∗ **_lcObj**

7.76.1 Detailed Description

template<class T>class UTIL::LCObjectHandle< T >

Template handle class for LCObjects.

Author

gaede

Version

Id:

LCObjectHandle.h,v 1.4 2005-02-03 14:11:00 gaede Exp

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

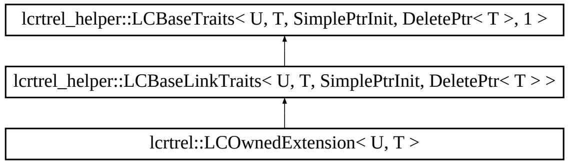
- UTIL/LCObjectHandle.h

7.77 lcrtrel::LCOwnedExtension< U, T > Class Template Reference

Simple Extension - pointer to an object of type T where the ownership is taken over by the object holding the extension, i.e.

```
#include <LCRTRelations.h>
```

Inheritance diagram for lcrtrel::LCOwnedExtension< U, T >:



Additional Inherited Members

7.77.1 Detailed Description

template<class U, typename T>class lcrtrel::LCOwnedExtension< U, T >

Simple Extension - pointer to an object of type T where the ownership is taken over by the object holding the extension, i.e.

it deletes the object when itself is deleted. The class U needs to be the subclass type, e.g.

struct MyAttributes : public LCExtension<MyAttributes,Attributes> {} ;

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

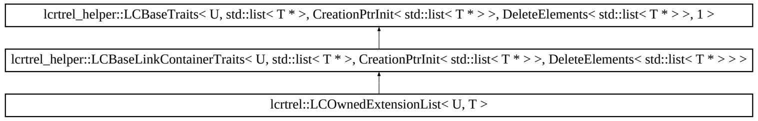
- LCRTRelations.h

7.78 lcrtrel::LCOwnedExtensionList< U, T > Class Template Reference

Extension list holding pointers to objects of type T - ownership of the objects is taken, i.e.

```
#include <LCRTRelations.h>
```

Inheritance diagram for lcrtrel::LCOwnedExtensionList< U, T >:



Additional Inherited Members

7.78.1 Detailed Description

template<class U, class T>class lcrtrel::LCOwnedExtensionList< U, T >

Extension list holding pointers to objects of type T - ownership of the objects is taken, i.e.

all objects pointed to in the vector are deleted when the object itself is deleted.

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

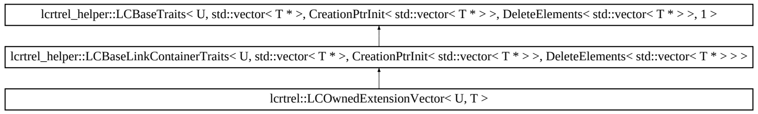
- LCRTRelations.h

7.79 lcrtrel::LCOwnedExtensionVector< U, T > Class Template Reference

Extension vector holding pointers to objects of type T - ownership of the objects is taken, i.e.

```
#include <LCRTRelations.h>
```

Inheritance diagram for lcrtrel::LCOwnedExtensionVector< U, T >:



Additional Inherited Members

7.79.1 Detailed Description

template<class U, class T>class lcrtrel::LCOwnedExtensionVector< U, T >

Extension vector holding pointers to objects of type T - ownership of the objects is taken, i.e.

all objects pointed to in the vector are deleted when the object itself is deleted.

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

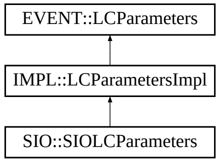
- LCRTRelations.h

7.80 EVENT::LCParameters Class Reference

Simple interface to store generic named parameters of type int, float and string.

```
#include <pre-generated/EVENT/LCParameters.h>
```

Inheritance diagram for EVENT::LCParameters:



Public Member Functions

- virtual ~**LCParameters** ()
  Destructor.

- virtual int **getIntVal** (const std::string &key) const =0

    Returns the first integer value for the given key.
- virtual float **getFloatVal** (const std::string &key) const =0

    Returns the first float value for the given key.
- virtual const std::string & **getStringVal** (const std::string &key) const =0

    Returns the first string value for the given key.
- virtual **IntVec** & **getIntVals** (const std::string &key, **IntVec** &values) const =0

    Adds all integer values for the given key to values.
- virtual **FloatVec** & **getFloatVals** (const std::string &key, **FloatVec** &values) const =0

    Adds all float values for the given key to values.
- virtual **StringVec** & **getStringVals** (const std::string &key, **StringVec** &values) const =0

    Adds all float values for the given key to values.
- virtual const **StringVec** & **getIntKeys** (**StringVec** &keys) const =0

    Returns a list of all keys of integer parameters.
- virtual const **StringVec** & **getFloatKeys** (**StringVec** &keys) const =0

    Returns a list of all keys of float parameters.
- virtual const **StringVec** & **getStringKeys** (**StringVec** &keys) const =0

    Returns a list of all keys of string parameters.
- virtual int **getNInt** (const std::string &key) const =0

    The number of integer values stored for this key.
- virtual int **getNFloat** (const std::string &key) const =0

    The number of float values stored for this key.
- virtual int **getNString** (const std::string &key) const =0

    The number of string values stored for this key.
- virtual void **setValue** (const std::string &key, int value)=0

    Set integer value for the given key.
- virtual void **setValue** (const std::string &key, float value)=0

    Set float value for the given key.
- virtual void **setValue** (const std::string &key, const std::string &value)=0

    Set string value for the given key.
- virtual void **setValues** (const std::string &key, const **IntVec** &values)=0

    Set integer values for the given key.
- virtual void **setValues** (const std::string &key, const **FloatVec** &values)=0

    Set float values for the given key.
- virtual void **setValues** (const std::string &key, const **StringVec** &values)=0

    Set string values for the given key.

7.80.1 Detailed Description

Simple interface to store generic named parameters of type int, float and string.

It can be used to store (user) meta data that is run, event or collection dependent.

Author

gaede

Version

Id:

LCParameters.aid,v 1.3 2009-11-20 11:20:20 gaede Exp

See Also

LCRunHeader.parameters() (p. 145)
LCEvent.parameters() (p. 92)
LCCollection.parameters() (p. 86)

7.80.2 Member Function Documentation

7.80.2.1 virtual FloatVec& EVENT::LCParameters::getFloatVals (const std::string & *key*, FloatVec & *values*) const `[pure virtual]`

Adds all float values for the given key to values.

Returns a reference to values for convenience.

Referenced by UTIL::LCTOOLS::printParameters(), and SIO::SIOLCParameters::write().

7.80.2.2 virtual IntVec& EVENT::LCParameters::getIntVals (const std::string & *key*, IntVec & *values*) const `[pure virtual]`

Adds all integer values for the given key to values.

Returns a reference to values for convenience.

Referenced by UTIL::IndexMap::IndexMap(), UTIL::LCTOOLS::printParameters(), and SIO::SIOLCParameters-::write().

7.80.2.3 virtual StringVec& EVENT::LCParameters::getStringVals (const std::string & *key*, StringVec & *values*) const `[pure virtual]`

Adds all float values for the given key to values.

Returns a reference to values for convenience.

Referenced by UTIL::IndexMap::IndexMap(), SIO::SIOVertexHandler::init(), UTIL::LCTOOLS::printParameters(), and SIO::SIOLCParameters::write().

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

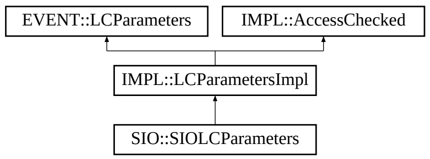
- LCParameters.h

7.81 IMPL::LCParametersImpl Class Reference

Implementation of Simple interface to store generic named parameters of type int, float and string.

`#include <IMPL/LCParametersImpl.h>`

Inheritance diagram for IMPL::LCParametersImpl:



Public Member Functions

- virtual ∼**LCParametersImpl** ()
  Destructor.
- virtual int **getIntVal** (const std::string &key) const
  Returns the first integer value for the given key.
- virtual float **getFloatVal** (const std::string &key) const
  Returns the first float value for the given key.
- virtual const std::string & **getStringVal** (const std::string &key) const
  Returns the first string value for the given key.
- virtual **EVENT::IntVec** & **getIntVals** (const std::string &key, **EVENT::IntVec** &values) const
  Adds all integer values for the given key to values.
- virtual **EVENT::FloatVec** & **getFloatVals** (const std::string &key, **EVENT::FloatVec** &values) const
  Adds all float values for the given key to values.
- virtual **EVENT::StringVec** & **getStringVals** (const std::string &key, **EVENT::StringVec** &values) const
  Adds all float values for the given key to values.
- virtual const **EVENT::StringVec** & **getIntKeys** (**EVENT::StringVec** &keys) const
  Returns a list of all keys of integer parameters.
- virtual const **EVENT::StringVec** & **getFloatKeys** (**EVENT::StringVec** &keys) const
  Returns a list of all keys of float parameters.
- virtual const **EVENT::StringVec** & **getStringKeys** (**EVENT::StringVec** &keys) const
  Returns a list of all keys of string parameters.
- virtual int **getNInt** (const std::string &key) const
  The number of integer values stored for this key.
- virtual int **getNFloat** (const std::string &key) const
  The number of float values stored for this key.
- virtual int **getNString** (const std::string &key) const
  The number of string values stored for this key.
- virtual void **setValue** (const std::string &key, int value)
  Set integer value for the given key.
- virtual void **setValue** (const std::string &key, float value)
  Set float value for the given key.
- virtual void **setValue** (const std::string &key, const std::string &value)
  Set string value for the given key.
- virtual void **setValues** (const std::string &key, const **EVENT::IntVec** &values)
  Set integer values for the given key.
- virtual void **setValues** (const std::string &key, const **EVENT::FloatVec** &values)
  Set float values for the given key.
- virtual void **setValues** (const std::string &key, const **EVENT::StringVec** &values)
  Set string values for the given key.

Protected Attributes

- IntMap **_intMap**
- FloatMap **_floatMap**
- StringMap **_stringMap**

Friends

- class **LCRunHeaderImpl**
- class **LCEventImpl**
- class **LCCollectionVec**

Additional Inherited Members

7.81.1 Detailed Description

Implementation of Simple interface to store generic named parameters of type int, float and string.

It can be used to store (user) meta data that is run, event or collection dependent.

Author

gaede

Version

Jun 23, 2004

See Also

LCRunHeader.parameters()
LCEvent.parameters()
LCCollection.parameters()

7.81.2 Member Function Documentation

7.81.2.1 FloatVec & IMPL::LCParametersImpl::getFloatVals (const std::string & *key*, EVENT::FloatVec & *values*) const `[virtual]`

Adds all float values for the given key to values.

Returns a reference to values for convenience.

7.81.2.2 IntVec & IMPL::LCParametersImpl::getIntVals (const std::string & *key*, EVENT::IntVec & *values*) const `[virtual]`

Adds all integer values for the given key to values.

Returns a reference to values for convenience.

7.81.2.3 StringVec & IMPL::LCParametersImpl::getStringVals (const std::string & *key*, EVENT::StringVec & *values*) const `[virtual]`

Adds all float values for the given key to values.

Returns a reference to values for convenience.

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

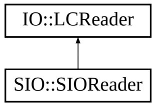
- LCParametersImpl.h
- LCParametersImpl.cc

7.82 IO::LCReader Class Reference

Interface for reading data from LCIO.

`#include <pre-generated/IO/LCReader.h>`

Inheritance diagram for IO::LCReader:



Public Member Functions

- virtual ∼**LCReader** ()
  Destructor.
- virtual void **open** (const std::string &filename)=0 throw (IOException, std::exception)
  Opens a file for reading (read-only).
- virtual void **open** (const std::vector< std::string > &filenames)=0 throw (IOException, std::exception)
  Opens a list of files for reading (read-only).
- virtual **EVENT::LCRunHeader** ∗ **readNextRunHeader** ()=0 throw (IOException, std::exception)
  Reads the next run header from the file.
- virtual **EVENT::LCRunHeader** ∗ **readNextRunHeader** (int accessMode)=0 throw (IOException, std::exception)
  *Same as **readNextRunHeader()** (p. 134) but allows to set the access mode LCIO::READ_ONLY (default) or LCIO::Update.*
- virtual **EVENT::LCEvent** ∗ **readNextEvent** ()=0 throw (IOException, std::exception)
  Reads the next event from the file.
- virtual **EVENT::LCEvent** ∗ **readNextEvent** (int accessMode)=0 throw (IOException, std::exception)
  *Same as **readNextEvent()** (p. 134) but allows to set the access mode LCIO::READ_ONLY (default) or LCIO::Update.*
- virtual int **getNumberOfEvents** ()=0 throw (IOException, std::exception)
  Return the number of events in the file - the file has to be open.
- virtual int **getNumberOfRuns** ()=0 throw (IOException, std::exception)
  Return the number of runs (run headers) in the file - the file has to be open.
- virtual void **getRuns** (**EVENT::IntVec** &runs)=0
  Return the run numbers of the runs (run headers) in the file - the file has to be open.
- virtual void **getEvents** (**EVENT::IntVec** &events)=0
  Return the run and event numbers of the events in the file - the file has to be open.
- virtual void **setReadCollectionNames** (const std::vector< std::string > &colnames)=0
  Limit the collection names that are going to be read to the subset given in the vector - all other collection will be ignored.
- virtual void **skipNEvents** (int n)=0 throw (IOException, std::exception)
  Skips the next n events from the current position.
- virtual **EVENT::LCRunHeader** ∗ **readRunHeader** (int runNumber)=0 throw (IOException, std::exception)
  Reads the specified runHeader from file.
- virtual **EVENT::LCRunHeader** ∗ **readRunHeader** (int runNumber, int accessMode)=0 throw (IOException, std::exception)
  *Same as LCEvent∗ **readRunHeader(int runNumber)** (p. 135) allowing to set the access mode LCIO::READ_ONLY (default) or LCIO::Update.*
- virtual **EVENT::LCEvent** ∗ **readEvent** (int runNumber, int evtNumber)=0 throw (IOException, std::exception)
  Reads the specified event from file.
- virtual **EVENT::LCEvent** ∗ **readEvent** (int runNumber, int evtNumber, int accessMode)=0 throw (IOException, std::exception)
  *Same as LCEvent∗ **readEvent(int runNumber, int evtNumber)** (p. 134) allowing to set the access mode LCIO::READ_ONLY (default) or LCIO::Update.*
- virtual void **close** ()=0 throw (IOException, std::exception)
  Closes the output file/stream etc.

- virtual void **registerLCEventListener** (**LCEventListener** ∗ls)=0
  Registers a listener for reading LCEvents from a stream.
- virtual void **removeLCEventListener** (**LCEventListener** ∗ls)=0
  Remove a listener for reading LCEvents from a stream.
- virtual void **registerLCRunListener** (**LCRunListener** ∗ls)=0
  Registers a listener for reading LCEventsLCRunHeaders from a stream.
- virtual void **removeLCRunListener** (**LCRunListener** ∗ls)=0
  Remove a listener for reading LCRunHeaders from a stream.
- virtual void **readStream** ()=0 throw (IOException, std::exception)
  Reads the input stream and notifies registered listeners according to the object type found in the stream.
- virtual void **readStream** (int maxRecord)=0 throw (IOException, std::exception)
  Reads maxRecord from the input stream and notifies registered listeners according to the object type found in the stream.

Static Public Attributes

- static const int **directAccess** = 0x00000001 << 0
  *Configuration flags for the **LCReader** (p. 130) instance - to be used with LCFactory::createLCReader().*

7.82.1 Detailed Description

Interface for reading data from LCIO.

Use LCFactory to instantiate a corresponding **LCWriter** (p. 166) object for the output format at hand (**SIO** (p. 30) only, so far).

See Also

LCFactory

Author

gaede

Version

Mar 4, 2003

7.82.2 Member Function Documentation

7.82.2.1 virtual void IO::LCReader::close ( ) throw IOException, std::exception) [pure virtual]

Closes the output file/stream etc.

Exceptions

IOException *(p. 79)*	

Implemented in **SIO::SIOReader** (p. 236).

7.82.2.2 virtual void IO::LCReader::getEvents (EVENT::IntVec & *events*) [pure virtual]

Return the run and event numbers of the events in the file - the file has to be open.

In case several input files are specified in the **open()** (p. 133) method - the run and event numbers of the events in the file that is currently open are returned. The size of the vector events will be twice the number of events, where i-th run number will be in events[2∗i] and the i-th event number in events[2∗i+].

Implemented in **SIO::SIOReader** (p. 236).

7.82.2.3 virtual int IO::LCReader::getNumberOfEvents ( ) throw IOException, std::exception) [pure virtual]

Return the number of events in the file - the file has to be open.

In case several input files are specified in the **open()** (p. 133) method - the number of events in the file that is currently open is returned.

Implemented in **SIO::SIOReader** (p. 236).

7.82.2.4 virtual int IO::LCReader::getNumberOfRuns ( ) throw IOException, std::exception) [pure virtual]

Return the number of runs (run headers) in the file - the file has to be open.

In case several input files are specified in the **open()** (p. 133) method - the number of runs (run headers) in the file that is currently open is returned.

Implemented in **SIO::SIOReader** (p. 236).

7.82.2.5 virtual void IO::LCReader::getRuns (EVENT::IntVec & *runs*) [pure virtual]

Return the run numbers of the runs (run headers) in the file - the file has to be open.

In case several input files are specified in the **open()** (p. 133) method - the run numbers of the runs (run headers) in the file that is currently open is returned.

Implemented in **SIO::SIOReader** (p. 237).

7.82.2.6 virtual void IO::LCReader::open (const std::string & *filename*) throw IOException, std::exception) [pure virtual]

Opens a file for reading (read-only).

Exceptions

IOException *(p. 79)*	

Implemented in **SIO::SIOReader** (p. 237).

7.82.2.7 virtual void IO::LCReader::open (const std::vector< std::string > & *filenames*) throw IOException, std::exception) [pure virtual]

Opens a list of files for reading (read-only).

All subsequent read operations will operate on the list, i.e. if an EOF is encountered the next file in the list will be opened and read transparently to the user.

Exceptions

IOException *(p. 79)*	

Implemented in **SIO::SIOReader** (p. 237).

7.82.2.8 virtual EVENT::LCEvent∗ IO::LCReader::readEvent (int *runNumber,* int *evtNumber*) throw IOException, std::exception) [pure virtual]

Reads the specified event from file.

Returns NULL if the specified event hasn't been found in the file.

Exceptions

IOException *(p. 79)*	

Implemented in **SIO::SIOReader** (p. 237).

7.82.2.9 virtual EVENT::LCEvent∗ IO::LCReader::readEvent (int *runNumber,* int *evtNumber,* int *accessMode*) throw IOException, std::exception) [pure virtual]

Same as LCEvent∗ **readEvent(int runNumber, int evtNumber)** (p. 134) allowing to set the access mode LCIO::R-EAD_ONLY (default) or LCIO::Update.

Exceptions

IOException *(p. 79)*	

Implemented in **SIO::SIOReader** (p. 238).

7.82.2.10 virtual EVENT::LCEvent∗ IO::LCReader::readNextEvent ( ) throw IOException, std::exception) [pure virtual]

Reads the next event from the file.

Returns NULL if 'EOF' read.

Exceptions

IOException *(p. 79)*	

Implemented in **SIO::SIOReader** (p. 239).

7.82.2.11 virtual EVENT::LCEvent∗ IO::LCReader::readNextEvent (int *accessMode*) throw IOException, std::exception) [pure virtual]

Same as **readNextEvent()** (p. 134) but allows to set the access mode LCIO::READ_ONLY (default) or LCIO::-Update.

Exceptions

IOException *(p. 79)*	

Implemented in **SIO::SIOReader** (p. 239).

7.82.2.12 virtual EVENT::LCRunHeader∗ IO::LCReader::readNextRunHeader ( ) throw IOException, std::exception) [pure virtual]

Reads the next run header from the file.

Returns NULL if 'EOF' read.

Exceptions

IOException *(p. 79)*	

Implemented in **SIO::SIOReader** (p. 239).

7.82.2.13 virtual EVENT::LCRunHeader∗ IO::LCReader::readNextRunHeader (int *accessMode*) throw IOException, std::exception) [pure virtual]

Same as **readNextRunHeader()** (p. 134) but allows to set the access mode LCIO::READ_ONLY (default) or LCIO-::Update.

Exceptions

IOException *(p. 79)*	

Implemented in **SIO::SIOReader** (p. 239).

7.82.2.14 virtual EVENT::LCRunHeader∗ IO::LCReader::readRunHeader (int *runNumber*) throw IOException, std::exception) [pure virtual]

Reads the specified runHeader from file.

Returns NULL if the specified runHeader hasn't been found in the file.

Exceptions

IOException *(p. 79)*	

Implemented in **SIO::SIOReader** (p. 239).

7.82.2.15 virtual EVENT::LCRunHeader∗ IO::LCReader::readRunHeader (int *runNumber,* int *accessMode*) throw IOException, std::exception) [pure virtual]

Same as LCEvent∗ **readRunHeader(int runNumber)** (p. 135) allowing to set the access mode LCIO::READ_ONLY (default) or LCIO::Update.

Exceptions

IOException *(p. 79)*	

Implemented in **SIO::SIOReader** (p. 240).

7.82.2.16 virtual void IO::LCReader::readStream ( ) throw IOException, std::exception) [pure virtual]

Reads the input stream and notifies registered listeners according to the object type found in the stream.

Exceptions

IOException *(p. 79)*	

Implemented in **SIO::SIOReader** (p. 240).

7.82.2.17 virtual void IO::LCReader::readStream (int *maxRecord*) throw IOException, std::exception) [pure virtual]

Reads maxRecord from the input stream and notifies registered listeners according to the object type found in the stream.

An exception is thrown if less than maxRecord where read from the file.

Exceptions

IOException *(p. 79)*	

Implemented in **SIO::SIOReader** (p. 240).

7.82.2.18 virtual void IO::LCReader::setReadCollectionNames (const std::vector< std::string > & *colnames*) [pure virtual]

Limit the collection names that are going to be read to the subset given in the vector - all other collection will be ignored.

This might improve the reading performance considerably in cases where only a small subset of the collections in the event is needed.

Implemented in **SIO::SIOReader** (p. 240).

7.82.2.19 virtual void IO::LCReader::skipNEvents (int *n*) throw IOException, std::exception) [pure virtual]

Skips the next n events from the current position.

Exceptions

IOException *(p. 79)*	

Implemented in **SIO::SIOReader** (p. 240).

7.82.3 Member Data Documentation

7.82.3.1 const int IO::LCReader::directAccess = 0x00000001 << 0 [static]

Configuration flags for the **LCReader** (p. 130) instance - to be used with LCFactory::createLCReader().

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

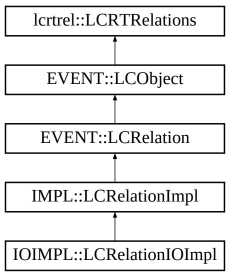
- LCReader.h

7.83 EVENT::LCRelation Class Reference

A single weighted relationship between two LCObjects.

`#include <pre-generated/EVENT/LCRelation.h>`

Inheritance diagram for EVENT::LCRelation:



Public Types

- typedef **LCRelation lcobject_type**

    *Useful typedef for template programming with **LCIO** (p. 115).*

Public Member Functions

- virtual ∼**LCRelation** ()

    Destructor.
- virtual **LCObject** ∗ **getFrom** () const =0

    The 'from' object of the given relation.
- virtual **LCObject** ∗ **getTo** () const =0

    The 'to' object of the given relation.
- virtual float **getWeight** () const =0

    The weight of the given relation - only if collection flag bit LCIO::LCREL_WEIGHTED is set.

Additional Inherited Members

7.83.1 Detailed Description

A single weighted relationship between two LCObjects.

Stored in an **LCCollection** (p. 83) like any other **LCObject** (p. 123). Use inplementations of LCRelationNavigator to navigate the relation efficiently. the types of the objects is decoded in the collection paramters 'RelationFromType' and 'RelationToType'. In order to store weights != 1. with the relation set the collection flag bit LCIO::LCREL_WEIGHTED, this is done automatically in **UTIL::LCRelationNavigator** (p. 139) if a weight != 1. exists.

Author

gaede

Version

Id:

LCRelation.aid,v 1.7 2006-09-14 10:16:11 gaede Exp

See Also

LCRelationNavigator

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

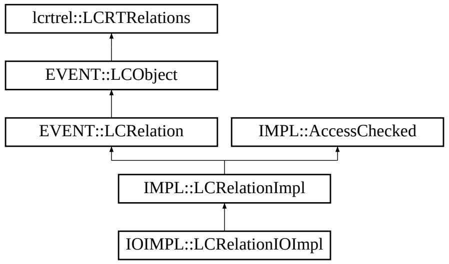
- LCRelation.h

7.84 IMPL::LCRelationImpl Class Reference

Implementation of the LCRelation.

`#include <IMPL/LCRelationImpl.h>`

Inheritance diagram for IMPL::LCRelationImpl:



Public Member Functions

- **LCRelationImpl** (**EVENT::LCObject** ∗from, **EVENT::LCObject** ∗to, float weight=1.0f)
- int **id** () const

    Returns an object id for internal (debugging) use in LCIO.
- **EVENT::LCObject** ∗ **getFrom** () const

    The 'from' object of the given relation.
- **EVENT::LCObject** ∗ **getTo** () const

    The 'to' object of the given relation.
- float **getWeight** () const

    The weight of the given relation - only if collection flag bit LCIO::LCREL_WEIGHTED is set.
- void **setFrom** (**EVENT::LCObject** ∗from)
- void **setTo** (**EVENT::LCObject** ∗to)
- void **setWeight** (float weight)

Protected Attributes

- **EVENT::LCObject** ∗ **_from**
- **EVENT::LCObject** ∗ **_to**
- float **_weight**

Additional Inherited Members

7.84.1 Detailed Description

Implementation of the LCRelation.

Author

gaede

Version

id:

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

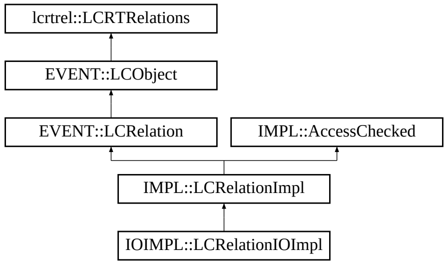
- LCRelationImpl.h

7.85 IOIMPL::LCRelationIOImpl Class Reference

Adding stuff needed for io (friend declarations, etc.)

`#include <IOIMPL/LCRelationIOImpl.h>`

Inheritance diagram for IOIMPL::LCRelationIOImpl:



Public Member Functions

- **LCRelationIOImpl** (**EVENT::LCObject** ∗from, **EVENT::LCObject** ∗to, float weight=1.0f)

Friends

- class **SIO::SIOLCRelationHandler**

Additional Inherited Members

7.85.1 Detailed Description

Adding stuff needed for io (friend declarations, etc.)

Author

gaede

Version

Id:

LCRelationIOImpl.h (p. **??**),v 1.4 2005-04-15 08:37:40 gaede Exp

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

- LCRelationIOImpl.h

7.86 UTIL::LCRelationNavigator Class Reference

The **LCRelationNavigator** (p. 139) makes repeated lookup of relations more conveneient and efficient.

`#include <UTIL/LCRelationNavigator.h>`

Public Member Functions

- **LCRelationNavigator** (const std::string &fromType, const std::string &toType)

  Default constructor.
- **LCRelationNavigator** (const **EVENT::LCCollection** ∗col)

  Create the navigator object from an existing collection of relations.
- virtual ∼**LCRelationNavigator** ()

  Destructor.
- virtual const std::string & **getFromType** () const

  The type of the 'from' objects in this relation.
- virtual const std::string & **getToType** () const

  The type of the 'to' objects in this relation.
- virtual const **EVENT::LCObjectVec** & **getRelatedToObjects** (**EVENT::LCObject** ∗from) const

  All objects that the given from-object is related to.
- virtual const **EVENT::LCObjectVec** & **getRelatedFromObjects** (**EVENT::LCObject** ∗to) const

  All from-objects related to the given object (the inverse relationship).
- virtual const **EVENT::FloatVec** & **getRelatedToWeights** (**EVENT::LCObject** ∗from) const

  The weights of the relations returned by a call to getRelatedToObjects(from).
- virtual const **EVENT::FloatVec** & **getRelatedFromWeights** (**EVENT::LCObject** ∗to) const

  The weights of the relations returned by a call to getRelatedFromObjects(to).
- virtual void **addRelation** (**EVENT::LCObject** ∗from, **EVENT::LCObject** ∗to, float weight=1.0)

  Adds a relation.
- virtual void **removeRelation** (**EVENT::LCObject** ∗from, **EVENT::LCObject** ∗to)

  Remove a given relation.
- virtual **EVENT::LCCollection** ∗ **createLCCollection** ()

  Remove a given relation.

Protected Member Functions

- virtual void **initialize** (const **EVENT::LCCollection** ∗col)
- void **removeRelation** (**EVENT::LCObject** ∗from, **EVENT::LCObject** ∗to, RelMap &map)
- void **addRelation** (**EVENT::LCObject** ∗from, **EVENT::LCObject** ∗to, float weight, RelMap &map)

Protected Attributes

- RelMap **_map**
- RelMap **_rMap**
- std::string **_from**
- std::string **_to**

7.86.1 Detailed Description

The **LCRelationNavigator** (p. 139) makes repeated lookup of relations more conveneient and efficient.

The relations are treated symmetrical, i.e. lookup of relations is equally efficient and fast for either direction (from-to and to-from) at the price of a slower (by a factor of ∼2) modification speed.

Author

gaede

Version

Id:

LCRelationNavigator.h (p. **??**),v 1.2 2004-09-06 14:35:51 gaede Exp

7.86.2 Member Function Documentation

7.86.2.1 void UTIL::LCRelationNavigator::addRelation (EVENT::LCObject ∗ *from*, EVENT::LCObject ∗ *to*, float *weight* = 1.0) [virtual]

Adds a relation.

If there is already an existing relation between the two given objects the weight (or default weight 1.0) is added to that relationship's weight.

References addRelation().

Referenced by addRelation().

7.86.2.2 EVENT::LCCollection ∗ UTIL::LCRelationNavigator::createLCCollection ( ) [virtual]

Remove a given relation.

To reduce the weight of the relationship, call addRelation(from, to, weight) with weight<0.

References IMPL::LCCollectionVec::addElement(), IMPL::LCFlagImpl::getFlag(), getFromType(), getToType(), IMPL::LCCollectionVec::parameters(), IMPL::LCFlagImpl::setBit(), IMPL::LCCollectionVec::setFlag(), and EVENT::LCParameters::setValue().

7.86.2.3 const EVENT::LCObjectVec & UTIL::LCRelationNavigator::getRelatedFromObjects (EVENT::LCObject ∗ *to*) const [virtual]

All from-objects related to the given object (the inverse relationship).

LCObjects are of type **getFromType()** (p. 140).

7.86.2.4 const EVENT::FloatVec & UTIL::LCRelationNavigator::getRelatedFromWeights (EVENT::LCObject ∗ *to*) const [virtual]

The weights of the relations returned by a call to getRelatedFromObjects(to).

See Also

getRelatedFromObjects (p. 141)

7.86.2.5 const EVENT::LCObjectVec & UTIL::LCRelationNavigator::getRelatedToObjects (EVENT::LCObject ∗ *from*) const [virtual]

All objects that the given from-object is related to.

LCObjects are of type **getToType()** (p. 140).

7.86.2.6 const EVENT::FloatVec & UTIL::LCRelationNavigator::getRelatedToWeights (EVENT::LCObject ∗ *from*) const [virtual]

The weights of the relations returned by a call to getRelatedToObjects(from).

See Also

getRelatedToObjects (p. 141)

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

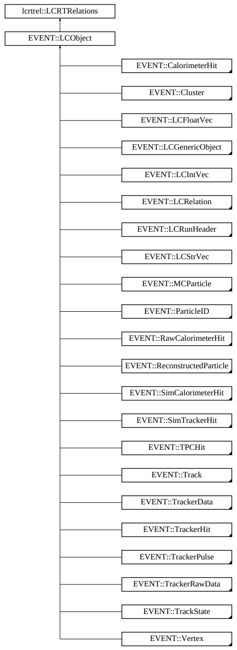
- LCRelationNavigator.h
- LCRelationNavigator.cc

7.87 lcrtrel::LCRTRelations Class Reference

Base class that provides run time (user) extensions and relation between objects.

`#include <LCRTRelations.h>`

Inheritance diagram for lcrtrel::LCRTRelations:



Public Member Functions

- template<class V >
  V::ext_type **ext** ()

  *Provides access to an extension object - the type and ownership is defined by the class V which should be a subtype of **LCExtension** (p. 99), **LCOwnedExtension** (p. 125), **LCExtensionVector** (p. 100), **LCExtensionList** (p. 100),...*
- template<class V >
  const V::ext_type **ext** () const
- template<class V >
  V::rel_type **rel** ()

*Provides read access to relations - the object types and their connectivity are defined by the class V which has to be a subtype of either **LC1To1Relation** (p. 80), **LC1ToNRelation** (p. 80) or **LCNToNRelation** (p. 122).*

Protected Member Functions

- template<class V >
  V::ptr & **ptr** () const

  Returns the reference to the pointer to the extension/relation object.

Friends

- template<class R >
  void **set_relation** (typename R::from::obj_ptr f, typename R::to::obj_ptr t)

  Set the 1-to-1 relation between two objects - prexisting inconsistent relations involving the two objects are deleted to enforce a consistent set of from-to relations.
- template<class R >
  void **unset_relation** (typename R::from::obj_ptr f)

  Unset the 1-to-1 relation from f.
- template<class R >
  void **add_relation** (typename R::from::obj_ptr f, typename R::to::obj_ptr t)

  Add a link from f to t to an N-to-N relation ship.
- template<class R >
  void **remove_relation** (typename R::from::obj_ptr f, typename R::to::obj_ptr t)

  Remove the link from from f to t from the N-to-N relation ship.
- template<class R >
  void **remove_relations** (typename R::from::obj_ptr f)

  Removes all relations from the given object.
- template<class R >
  void **merge_relations** (typename R::from::obj_ptr f1, typename R::from::obj_ptr f2)

  Merge the relations from f2 to f1 - after this call f1 will hold all the relations and f2 will be empty.

7.87.1 Detailed Description

Base class that provides run time (user) extensions and relation between objects.

Every subclass object of **LCRTRelations** (p. 142) (and thus LCObbject) will automatically have the following functionality:

- extension of the object with arbitrary (even non-LCObject) classes
- extension of single objects or vectors, lists of objects
- optionally ownership is taken for extension objects (memory management)
- relations to other subclasses of **LCRTRelations** (p. 142):
  - one to one
  - one to many
  - many to many

The described functionality is provided through the two templated member functions:

ext<class V>() (p. 142)

rel<class V>() (p. 142)

the class V is a user defined so called traits class, that uniquely tags the extension/relationship and defines the types of the objects involved.

For extensions users have to subclass one of the following classes:

LCExtension (p. 99), **LCOwnedExtension** (p. 125), **LCIntExtension** (p. 113), **LCFloatExtension** (p. 105),

LCExtensionVector (p. 100), **LCExtensionList** (p. 100), **LCOwnedExtensionVector** (p. 126), **LCOwned-ExtensionList** (p. 125).

For example the following defines a user extension of a vector of strings that are owned by the object (i.e. deleted when the object is deleted):

struct ParticleIDs (p. 185) : LCOwnedExtensionVector<ParticleIDs, std::string> {};

(note: the first template parameter has to be the class itself !)

This extension can then be used anywhere in the following code for all LCObjects, e.g.:

MCParticle∗ mcp = dynamic_cast<MCParticle∗>(mcpcol->getElementAt(i)) ;

mcp->ext<ParticleIDs>() (p. 142)->push_back(new std::string("charged")) ;

mcp->ext<ParticleIDs>() (p. 142)->push_back(new std::string("hadron")) ;

mcp->ext<ParticleIDs>() (p. 142)->push_back(new std::string("pion")) ;

and be read out again:

ParticleIDs::ext_type pidv = mcp->ext<ParticleIDs>() (p. 142) ;

for(ParticleIDs::const_iterator ipid = pidv->begin() ; ipid != pidv->end(); ++ipid){

std::cout << ∗∗ipid << ", " ;

}

Similarily the following defines a one to many relationship between Tracks and Clusters:

struct TrkCluLink (p. 310) : LC1ToNRelation<TrkCluLink,Track,Cluster> {} ;

Relations are allways biderectional, i.e. there is a **from** and a **to** side. They can then be set and modified with the following functions:

set_relation() (p. 145), **unset_relation()** (p. 143), **add_relation()** (p. 143),

remove_relation() (p. 143), **remove_relations()** (p. 143), **merge_relations()** (p. 143).

For example:

Track∗ trk = dynamic_cast<Track∗> (trkcol->getElementAt(j)) ;

//...

Cluster∗ clu = dynamic_cast<Cluster∗> (clucol->getElementAt(k)) ;

add_relation<TrkCluLink>(trk ,clu);

The many side can then be read out with a const_iterator and the one side with a normal pointer:

Track∗ trk = clu->rel<TrkCluLink::from>() (p. 142) ;

//...

TrkCluLink::to::rel_type clulist = trk->rel<TrkCluLink::to>() (p. 142) ;

for(TrkCluLink::to::const_iterator iclu = clulist->begin() ; iclu != clulist->end() ; ++iclu){

Cluster∗ clu = ∗iclu ; // iterator is of type pointer to container element

std::cout << " assigned cluster with E = " << clu->getEnergy() << std::endl ;

}

More examples can be found in **$LCIO/src/cpp/src/EXAMPLES/lcrtrelations.cc**.

7.87.2 Friends And Related Function Documentation

7.87.2.1 template< class R > void set_relation (typename R::from::obj_ptr *f*, typename R::to::obj_ptr *t*) [friend]

Set the 1-to-1 relation between two objects - prexisting inconsistent relations involving the two objects are deleted to enforce a consistent set of from-to relations.

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

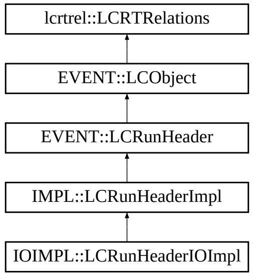
- LCRTRelations.h

7.88 EVENT::LCRunHeader Class Reference

Interface for the run header.

`#include <pre-generated/EVENT/LCRunHeader.h>`

Inheritance diagram for EVENT::LCRunHeader:



Public Types

- typedef **LCRunHeader lcobject_type**

  Useful typedef for template programming with ***LCIO*** *(*p. *115).*

Public Member Functions

- virtual ∼**LCRunHeader** ()

  Destructor.
- virtual int **getRunNumber** () const =0

  Returns the run number.
- virtual const std::string & **getDetectorName** () const =0

  Returns the name of the detector setup used in the simulation.
- virtual const std::string & **getDescription** () const =0

  Description of the simulation, physics channels etc.
- virtual const std::vector
  < std::string > ∗ **getActiveSubdetectors** () const =0

  Returns the names of the active subdetectors used in the simulation.
- virtual const **LCParameters** & **getParameters** () const =0

  Parameters defined for this run.
- virtual **LCParameters** & **parameters** ()=0

  Parameters defined for this run.

Additional Inherited Members

7.88.1 Detailed Description

Interface for the run header.

Author

gaede

Version

Id:

LCRunHeader.aid,v 1.8 2006-03-24 13:25:52 gaede Exp

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

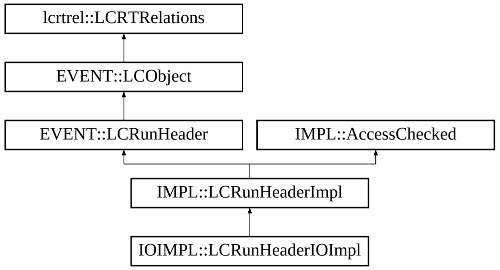
- LCRunHeader.h

7.89 IMPL::LCRunHeaderImpl Class Reference

Implementation of LCRunHeader.

`#include <IMPL/LCRunHeaderImpl.h>`

Inheritance diagram for IMPL::LCRunHeaderImpl:



Public Member Functions

- virtual ∼**LCRunHeaderImpl** ()

  Destructor.
- virtual int **id** () const

  Returns an object id for internal (debugging) use in LCIO.
- virtual int **getRunNumber** () const

  Returns the run number.
- virtual const std::string & **getDetectorName** () const

  Returns the name of the detector setup used in the simulation.
- virtual const std::string & **getDescription** () const

  Description of the simulation conditions, e.g.

- virtual const std::vector
  < std::string > ∗ **getActiveSubdetectors** () const
    Returns the names of the active subdetectors used in the simulation.
- virtual const **EVENT::LCParameters** & **getParameters** () const
    Parameters defined for this run.
- virtual **EVENT::LCParameters** & **parameters** ()
    Parameters defined for this run.
- void **setRunNumber** (int rn)
    Sets the run number.
- void **setDetectorName** (const std::string &dn)
    Sets the detector name.
- void **setDescription** (const std::string &dsc)
    Sets the description of the run.
- void **addActiveSubdetector** (const std::string &adn)
    Adds an active sub detector name.

Protected Member Functions

- virtual void **setReadOnly** (bool readOnly)

Protected Attributes

- int **_runNumber**
- std::string **_detectorName**
- std::string **_description**
- std::vector< std::string > **_activeSubdetectors**
- **LCParametersImpl _params**

Additional Inherited Members

7.89.1 Detailed Description

Implementation of LCRunHeader.

Author

gaede

Version

Mar 11, 2003

7.89.2 Member Function Documentation

7.89.2.1 const std::string & IMPL::LCRunHeaderImpl::getDescription ( ) const [virtual]

Description of the simulation conditions, e.g.

physics channels.

Implements **EVENT::LCRunHeader** (p. 145).

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

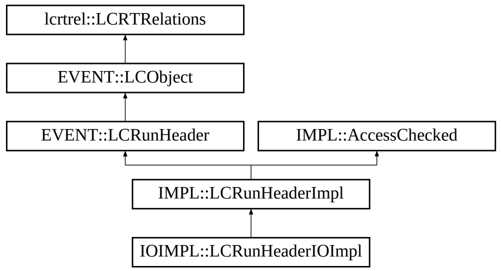
- LCRunHeaderImpl.h
- LCRunHeaderImpl.cc

7.90 IOIMPL::LCRunHeaderIOImpl Class Reference

Adding stuff needed for io (friend declarations, etc.)

`#include <IOIMPL/LCRunHeaderIOImpl.h>`

Inheritance diagram for IOIMPL::LCRunHeaderIOImpl:



Friends

- class **SIO::SIOReader**

Additional Inherited Members

7.90.1 Detailed Description

Adding stuff needed for io (friend declarations, etc.)

Author

gaede

Version

Sep 9, 2003

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

- LCRunHeaderIOImpl.h

7.91 IO::LCRunListener Class Reference

Listener for the occurence of LCRunHeader when reading a stream.

`#include <pre-generated/IO/LCRunListener.h>`

Public Member Functions

- virtual ∼**LCRunListener** ()
    Destructor.
- virtual void **processRunHeader** (**EVENT::LCRunHeader** ∗rh)=0

Call back for processing an LCRunHeader.
- virtual void **modifyRunHeader** (**EVENT::LCRunHeader** *rh)=0
  Call back for modifying an existing LCRunHeader.

7.91.1 Detailed Description

Listener for the occurence of LCRunHeader when reading a stream.

User classes should implement this interface for their analysis code and then register the listener with **LCReader** (p. 130).

Author

gaede

Version

Mar 11, 2003

See Also

LCRunHeader
LCReader (p. 130)

7.91.2 Member Function Documentation

7.91.2.1 virtual void IO::LCRunListener::processRunHeader (EVENT::LCRunHeader * *rh*) [pure virtual]

Call back for processing an LCRunHeader.

Access is 'read only'.

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

- LCRunListener.h

7.92 SIO::LCSIO Class Reference

Collection of static helper functions for reading and writing data with **SIO** (p. 30).

```
#include <SIO/LCSIO.h>
```

Static Public Member Functions

- static **SIORecords** & **records** ()
- static void **checkVersion** (int versionID)
  Checks the version of the file - oldefile (version < v00-08) are no longer supported.
- static unsigned int **read** (SIO_stream *stream, char **c, int *len=0)
  Read a string from the stream into a dummy buffer.
- static unsigned int **write** (SIO_stream *stream, int i)
  This version checks the versionId to be able to read 'old' files with trailing '\00' (version <= 00-02).
- static unsigned int **write** (SIO_stream *stream, unsigned int i)
  Write an unsigned int to the stream.
- static unsigned int **write** (SIO_stream *stream, **EVENT::long64** i)

Write a vector size to the stream (needed for AMD64 architectures)
- static unsigned int **write** (SIO_stream *stream, float f)
  Write a float to the stream.
- static unsigned int **write** (SIO_stream *stream, const std::string &s)
  Write a string to the stream.
- static std::string **getValidSIOName** (const std::string &aName)
  *Creates a valid **SIO** (p. 30) name (basically equivalent to a valid C++ name) by replacing every [.*
- static void **seekStream** (SIO_stream *stream, **EVENT::long64** pos)
  Seek stream to the given absolute position - if pos<0 from end of file.

Static Public Attributes

- static const char * **FILE_EXTENSION** =".slcio"
- static const bool **COMPRESSION** = true
  *the compression mode for **SIO** (p. 30)*

7.92.1 Detailed Description

Collection of static helper functions for reading and writing data with **SIO** (p. 30).

Could go to the SIO_functions class. Also defines some constant names for lcio-SIO.

Author

gaede

Version

Mar 6, 2003

7.92.2 Member Function Documentation

7.92.2.1 std::string SIO::LCSIO::getValidSIOName (const std::string & *aName*) [static]

Creates a valid **SIO** (p. 30) name (basically equivalent to a valid C++ name) by replacing every [.

], [] and [/] with underscores and removing all nonalphanumeric characters and adding an 'A' to the beginning, if necessary.

Referenced by SIO::SIOReader::open(), and SIO::SIOWriter::open().

7.92.2.2 unsigned int SIO::LCSIO::read (SIO_stream * *stream,* char ** *c,* int * *len* = 0) [static]

Read a string from the stream into a dummy buffer.

Warning the same buffer is used for each call. So the return value needs to be copied to its final memory destination. len holds the length of the string (can contain '\0').

7.92.2.3 void SIO::LCSIO::seekStream (SIO_stream * *stream,* EVENT::long64 *pos*) [static]

Seek stream to the given absolute position - if pos<0 from end of file.

Exceptions

IOException	

Referenced by SIO::LCIORandomAccessMgr::createEventMap(), SIO::SIOWriter::open(), SIO::LCIORandomAccessMgr::readLCIOIndexAt(), SIO::LCIORandomAccessMgr::readLCIORandomAccessAt(), and SIO::LCIORandomAccessMgr::recreateEventMap().

7.92.2.4 unsigned int SIO::LCSIO::write (SIO_stream ∗ *stream*, int *i*) [static]

This version checks the versionId to be able to read 'old' files with trailing '\00' (version <= 00-02).

Remove this method after a reasonable transition period...Write an int to the stream.

7.92.2.5 unsigned int SIO::LCSIO::write (SIO_stream ∗ *stream*, EVENT::long64 *i*) [static]

Write a vector size to the stream (needed for AMD64 architectures)

Write a 64bit integer to the stream

7.92.2.6 unsigned int SIO::LCSIO::write (SIO_stream ∗ *stream*, const std::string & *s*) [static]

Write a string to the stream.

Writes the length of the string first and adds '\0' to the end.

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

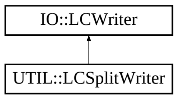
- LCSIO.h
- LCSIO.cc

7.93 UTIL::LCSplitWriter Class Reference

LCWriter wrapper that automatically splits files if a given number of bytes is exceeded.

`#include <UTIL/LCSplitWriter.h>`

Inheritance diagram for UTIL::LCSplitWriter:



Public Member Functions

- **LCSplitWriter** (**IO::LCWriter** ∗wrt, **EVENT::long64** maxBytes)
  The constructor.
- virtual ∼**LCSplitWriter** ()
  Destructor.
- virtual void **open** (const std::string &filename) throw (IO::IOException, std::exception)
  Opens a file for writing where the filename has to include the extension but not the number, e.g.
- virtual void **open** (const std::string &filename, int writeMode) throw (IO::IOException, std::exception)
  Not implemented - will throw an Exception if called.

- virtual void **writeRunHeader** (const **EVENT::LCRunHeader** ∗hdr) throw (IO::IOException, std::exception)
  Writes the given run header to file.
- virtual void **setCompressionLevel** (int level)
  Set CompressionLevel.
- virtual void **writeEvent** (const **EVENT::LCEvent** ∗evt) throw (IO::IOException, std::exception)
  Writes the given file to file.
- virtual void **close** () throw (IO::IOException, std::exception)
  Closes the output file/stream.
- virtual void **flush** () throw (IO::IOException, std::exception)
  Flushes the output file/stream.
- **EVENT::long64 fileSize** ()
  Return the current file size in bytes.

Protected Member Functions

- **EVENT::long64 file_size** (const char ∗fname)
  Helper function that returns the file size in bytes - o if the file does not exist.
- const std::string & **getFilename** ()
  Get the complete filename w/o extensiomn,e.g.
- void **setBaseFilename** (const std::string &filename)
  Helper function to determine base file name, i.e.
- std::string **getCountingString** (unsigned count)
  Returns the string representation of the file number, e.g.

Protected Attributes

- **IO::LCWriter** ∗ **_wrt**
- **EVENT::long64 _maxBytes**
- std::string **_baseFilename**
- std::string **_filename**
- std::string **_extension**
- unsigned **_count**
- unsigned **_lastCount**

7.93.1 Detailed Description

LCWriter wrapper that automatically splits files if a given number of bytes is exceeded.

File names are extended by a file number of the form .000 - .999 ...

Author

gaede

Version

Id:

LCSplitWriter.h (p. **??**),v 1.2 2007-11-09 21:00:55 gaede Exp

7.93.2 Constructor & Destructor Documentation

7.93.2.1 UTIL::LCSplitWriter::LCSplitWriter (IO::LCWriter ∗ *wrt*, EVENT::long64 *maxBytes*) [inline]

The constructor.

Takes a pointer to a valid LCWriter instance and the file size in bytes that causes a new file to be opened if exceeded.**example:**

// replace LCWriter∗ lcWrt = LCFactory::getInstance()->createLCWriter() ; // with LCSplitWriter∗ lcWrt = new LCSplitWriter (p. 151)(LCFactory::getInstance()->createLCWriter() , 2040109465) ;

to automatically split files after 1.9 GByte file size has been exceeded.

7.93.3 Member Function Documentation

7.93.3.1 void UTIL::LCSplitWriter::close ( ) throw IO::IOException, std::exception) [virtual]

Closes the output file/stream.

Exceptions

IO::IOException (p. 79)	

Implements **IO::LCWriter** (p. 167).

7.93.3.2 void UTIL::LCSplitWriter::flush ( ) throw IO::IOException, std::exception) [virtual]

Flushes the output file/stream.

Exceptions

IO::IOException (p. 79)	

Implements **IO::LCWriter** (p. 167).

7.93.3.3 std::string UTIL::LCSplitWriter::getCountingString (unsigned *count*) [protected]

Returns the string representation of the file number, e.g.

"007".

7.93.3.4 const std::string & UTIL::LCSplitWriter::getFilename ( ) [protected]

Get the complete filename w/o extensiomn,e.g.

MyFilename.007

7.93.3.5 void UTIL::LCSplitWriter::open (const std::string & *filename*) throw IO::IOException, std::exception) [virtual]

Opens a file for writing where the filename has to include the extension but not the number, e.g.

myfile.slcio. Note that this is different from the LCWriter specification.

Exceptions

IO::IOException (p. 79)	

Implements **IO::LCWriter** (p. 167).

7.93.3.6 void UTIL::LCSplitWriter::open (const std::string & *filename*, int *writeMode*) throw IO::IOException, std::exception) [virtual]

Not implemented - will throw an Exception if called.

Overwriting of or appending to split files is not straight forward. Pleas use the default write mode and remove exisiting files.

Exceptions

IO::IOException (p. 79)	

Implements **IO::LCWriter** (p. 168).

7.93.3.7 void UTIL::LCSplitWriter::setBaseFilename (const std::string & *filename*) [protected]

Helper function to determine base file name, i.e.

w/o extension.

7.93.3.8 virtual void UTIL::LCSplitWriter::setCompressionLevel (int *level*) [inline], [virtual]

Set CompressionLevel.

See Also

LCWriter::setCompressionLevel()

Implements **IO::LCWriter** (p. 168).

References IO::LCWriter::setCompressionLevel().

7.93.3.9 void UTIL::LCSplitWriter::writeEvent (const EVENT::LCEvent ∗ *evt*) throw IO::IOException, std::exception) [virtual]

Writes the given file to file.

Opens a new file if the given file size is already exceeded before the execution of the write access.

Exceptions

IO::IOException (p. 79)	

Implements **IO::LCWriter** (p. 168).

7.93.3.10 void UTIL::LCSplitWriter::writeRunHeader (const EVENT::LCRunHeader ∗ *hdr*) throw IO::IOException, std::exception) [virtual]

Writes the given run header to file.

Opens a new file if the given file size is already exceeded before the execution of the write access.

Exceptions

IO::IOException *(p. 79)*	

Implements **IO::LCWriter** (p. 168).

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

- LCSplitWriter.h
- LCSplitWriter.cc

7.94 UTIL::LCStdHepRdr Class Reference

Basic utility for reading a binary stdhep file and filling a LCCollectionVec with MCParticles containing the stdhep file information.

`#include <UTIL/LCStdHepRdr.h>`

Public Member Functions

- **LCStdHepRdr** (const char *evfile)
  Open the stdhep input file in the constructer.
- ∼**LCStdHepRdr** ()
  noop
- long **getNumberOfEvents** () const
  Get number of events in the stdhep file.
- long **getNumberOfTotalEventsExpected** () const
  Get total number of expected events in the whole set of stdhep files from which this stdhep file belongs to.
- **IMPL::LCCollectionVec** * **readEvent** ()
  Read an event and return an LCCollectionVec of MCParticles.
- void **updateNextEvent** (**IMPL::LCEventImpl** *evt, const char *colName=EVENT::LCIO::MCPARTICLE)
  Reads the next stdhep event and adds a new MCParticle collection to the the event with default name 'MCParticle'.
- void **printHeader** (std::ostream &os=std::cout)
  Print the file header to the given ostream.
- int **threeCharge** (int pdgID) const
  Return the charge of the particle times 3 - code copied from HepPDT package.

7.94.1 Detailed Description

Basic utility for reading a binary stdhep file and filling a LCCollectionVec with MCParticles containing the stdhep file information.

Author

cassell

Version

Id:

LCStdHepRdr.h (p. **??**),v 1.4 2007-11-12 16:39:04 gaede Exp

7.94.2 Member Function Documentation

7.94.2.1 long UTIL::LCStdHepRdr::getNumberOfEvents ( ) const [inline]

Get number of events in the stdhep file.

This number is read from the file header (no guarantee that it is correct)

7.94.2.2 long UTIL::LCStdHepRdr::getNumberOfTotalEventsExpected ( ) const [inline]

Get total number of expected events in the whole set of stdhep files from which this stdhep file belongs to.

This number is read from the file header (no guarantee that it is correct)

7.94.2.3 IMPL::LCCollectionVec * UTIL::LCStdHepRdr::readEvent ( )

Read an event and return an LCCollectionVec of MCParticles.

Deprecated please use updateEvent()

References IMPL::MCParticleImpl::addParent(), IMPL::MCParticleImpl::getDaughters(), IMPL::LCCollectionVec::getElementAt(), IMPL::MCParticleImpl::getGeneratorStatus(), IMPL::MCParticleImpl::getParents(), IMPL::MCParticleImpl::getPDG(), IMPL::LCCollectionVec::parameters(), IMPL::MCParticleImpl::setCharge(), IMPL::MCParticleImpl::setColorFlow(), IMPL::MCParticleImpl::setGeneratorStatus(), IMPL::MCParticleImpl::setMass(), IMPL::MCParticleImpl::setMomentum(), IMPL::MCParticleImpl::setPDG(), IMPL::MCParticleImpl::setSimulatorStatus(), IMPL::MCParticleImpl::setSpin(), IMPL::MCParticleImpl::setTime(), EVENT::LCParameters::setValue(), and IMPL::MCParticleImpl::setVertex().

7.94.2.4 int UTIL::LCStdHepRdr::threeCharge (int *pdgID*) const

Return the charge of the particle times 3 - code copied from HepPDT package.

PID digits (base 10) are: n nr nl nq1 nq2 nq3 nj The location enum provides a convenient index into the PID.

7.94.2.5 void UTIL::LCStdHepRdr::updateNextEvent (IMPL::LCEventImpl * *evt*, const char * *colName* = EVENT::LCIO::MCPARTICLE)

Reads the next stdhep event and adds a new MCParticle collection to the the event with default name 'MCParticle'.

Exceptions

IO::EndOfDataException *(p. 68)*	if no event in stdhep file

References IMPL::LCEventImpl::addCollection(), EVENT::LCParameters::getFloatVal(), EVENT::LCParameters::getIntVal(), IMPL::LCCollectionVec::getParameters(), IMPL::LCEventImpl::parameters(), EVENT::LCParameters::setValue(), and IMPL::LCEventImpl::setWeight().

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

- LCStdHepRdr.h
- LCStdHepRdr.cc

7.95 UTIL::LCStdHepRdrNew Class Reference

Basic utility for reading a binary stdhep file and filling a LCCollectionVec with MCParticles containing the stdhep file information.

`#include <UTIL/LCStdHepRdrNew.h>`

Public Member Functions

- **LCStdHepRdrNew** (const char ∗evfile)
  Open the stdhep input file in the constructer.
- ∼**LCStdHepRdrNew** ()
  noop
- long **getNumberOfEvents** () const
  Get number of events in the stdhep file.
- long **getNumberOfTotalEventsExpected** () const
  Get total number of expected events in the whole set of stdhep files from which this stdhep file belongs to.
- **IMPL::LCCollectionVec** ∗ **readEvent** ()
  Read an event and return an LCCollectionVec of MCParticles.
- void **updateNextEvent** (**IMPL::LCEventImpl** ∗evt, const char ∗colName=EVENT::LCIO::MCPARTICLE)
  Reads the next stdhep event and adds a new MCParticle collection to the the event with default name 'MCParticle'.
- void **printHeader** (std::ostream &os=std::cout)
  Print the file header to the given ostream.
- int **threeCharge** (int pdgID) const
  Return the charge of the particle times 3 - code copied from HepPDT package.
- void **setWriteEventNumber** (bool writeEventNumber)

7.95.1 Detailed Description

Basic utility for reading a binary stdhep file and filling a LCCollectionVec with MCParticles containing the stdhep file information.

Author

cassell

Version

Id:

LCStdHepRdr.h (p. **??**),v 1.4 2007-11-12 16:39:04 gaede Exp

7.95.2 Member Function Documentation

7.95.2.1 long UTIL::LCStdHepRdrNew::getNumberOfEvents ( ) const [inline]

Get number of events in the stdhep file.

This number is read from the file header (no guarantee that it is correct)

7.95.2.2 long UTIL::LCStdHepRdrNew::getNumberOfTotalEventsExpected ( ) const [inline]

Get total number of expected events in the whole set of stdhep files from which this stdhep file belongs to.

This number is read from the file header (no guarantee that it is correct)

7.95.2.3 IMPL::LCCollectionVec ∗ UTIL::LCStdHepRdrNew::readEvent ( )

Read an event and return an LCCollectionVec of MCParticles.

Deprecated please use updateEvent()

References IMPL::MCParticleImpl::addParent(), IMPL::MCParticleImpl::getDaughters(), IMPL::LCCollectionVec::getElementAt(), IMPL::MCParticleImpl::getGeneratorStatus(), IMPL::MCParticleImpl::getParents(), IMPL::MCParticleImpl::getPDG(), IMPL::LCCollectionVec::parameters(), IMPL::MCParticleImpl::setCharge(), IMPL::MCParticleImpl::setColorFlow(), IMPL::MCParticleImpl::setGeneratorStatus(), IMPL::MCParticleImpl::setMass(), IMPL::MCParticleImpl::setMomentum(), IMPL::MCParticleImpl::setPDG(), IMPL::MCParticleImpl::setSimulatorStatus(), IMPL::MCParticleImpl::setSpin(), IMPL::MCParticleImpl::setTime(), EVENT::LCParameters::setValue(), and IMPL::MCParticleImpl::setVertex().

7.95.2.4 int UTIL::LCStdHepRdrNew::threeCharge (int *pdgID*) const

Return the charge of the particle times 3 - code copied from HepPDT package.

PID digits (base 10) are: n nr nl nq1 nq2 nq3 nj The location enum provides a convenient index into the PID.

7.95.2.5 void UTIL::LCStdHepRdrNew::updateNextEvent (IMPL::LCEventImpl ∗ *evt*, const char ∗ *colName* = EVENT::LCIO::MCPARTICLE)

Reads the next stdhep event and adds a new MCParticle collection to the the event with default name 'MCParticle'.

Exceptions

IO::EndOfDataException *(p. 68)*	if no event in stdhep file

References IMPL::LCEventImpl::addCollection(), EVENT::LCParameters::getFloatVal(), EVENT::LCParameters::getIntVal(), IMPL::LCCollectionVec::getParameters(), IMPL::LCEventImpl::parameters(), IMPL::LCEventImpl::setEventNumber(), EVENT::LCParameters::setValue(), and IMPL::LCEventImpl::setWeight().

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

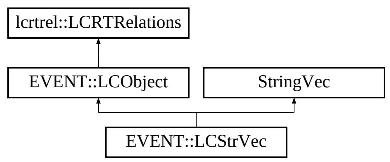
- LCStdHepRdrNew.h
- LCStdHepRdrNew.cc

7.96 EVENT::LCStrVec Class Reference

String vector used for user extensions.

`#include <pre-generated/EVENT/LCStrVec.h>`

Inheritance diagram for EVENT::LCStrVec:



Public Types

- typedef **LCStrVec lcobject_type**

Public Member Functions

- virtual ∼**LCStrVec** ()
  Destructor.
- int **id** () const
  Returns an object id for internal (debugging) use in ***LCIO*** *(p. 115).*

Protected Attributes

- **IMPL::AccessChecked _acc**

Additional Inherited Members

7.96.1 Detailed Description

String vector used for user extensions.

The cpp implementation is a std::vector<string>

Author

vogt

Version

Jun 18, 2004

See Also

LCObject (p. 123)

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

- pre-generated/EVENT/LCStrVec.h

7.97 UTIL::LCTime Class Reference

Helper class that allows to convert time stamps as defined in LCEvent::getTimeStamp() (ns since 1.1.1970 00:00:00 UTC) to ordinary date and time and vice versa.

```
#include <UTIL/LCTime.h>
```

Classes

- struct **CalendarTime**
  Helper struct that holds the calendar time.

Public Member Functions

- **LCTime** ()
  Init with current time (accuracy depends on OS/machine,etc.) in UTC/GMT.
- **LCTime** (**EVENT::long64** time)
  Init from 64 bit time stamp (ns since 1.1.1970 00:00:00).
- **LCTime** (int **year**, int **month**, int **day**, int **hour**=0, int **min**=0, int s=0)
  Init from date and time.
- **LCTime** (int **unixTime**)
  Init from unix time (s since 1.1.1970 00:00:00).
- int **unixTime** () const
  Unix time (s since 1.1.1970 00:00:00).
- **EVENT::long64 timeStamp** () const
  64bit time stamp as used in LCEvent::getTimestamp().
- bool **isLeapYear** (int **year**) const
  True if the year is a leap year in the Gregorian calendar:
  year is multiple of 4 and not multiple of 100
  or year is multiple of 400.
- int **daysInYear** (int **year**) const
  The number if days in the given year in the Gregorian calendar.
- int **daysInMonth** (int **month**, int **year**) const
  The number if days in the given month and year in the Gregorian calendar.
- std::string **getDateString** () const
  Date in human readable format, e.g.
- const **CalendarTime** & **calendarTime** () const
  Calendar time: year,month,day hour,min,sec and ns.
- int **year** () const
  Calendar time: year.
- int **month** () const
  Calendar time: month.
- int **day** () const
  Calendar time: day.
- int **hour** () const
  Calendar time: hour.
- int **min** () const
  Calendar time: min.
- int **sec** () const
  Calendar time: sec.
- int **ns** () const
  Calendar time: ns.
- void **operator+=** (**EVENT::long64** t)
  Add the specified number of ns to the time.

Static Public Member Functions

- static bool **test** (int nDates)
  Tests the ***LCTime*** *(p. 159) class with nDates random dates Returns true if successful - throws Exception in case of error.*

Protected Member Functions

- void **convertToCalTime** ()
- void **convertFromCalTime** ()

Protected Attributes

- **EVENT::long64 _t**
- **CalendarTime _d**

Static Protected Attributes

- static int **dpm** [13] = { 0,31,28,31,30,31,30,31,31,30,31,30,31 }

7.97.1 Detailed Description

Helper class that allows to convert time stamps as defined in LCEvent::getTimeStamp() (ns since 1.1.1970 00:00:00 UTC) to ordinary date and time and vice versa.

Dates and times are in UTC/GMT by default.

Negative time stamps are not supported and mapped to 0LL and 1.1.1970 00:00:00 UTC respectively.

Author

gaede

Version

Id:

LCTime.h (p. **??**),v 1.4 2005-04-22 15:39:01 gaede Exp

7.97.2 Member Function Documentation

7.97.2.1 int UTIL::LCTime::daysInMonth (int *month*, int *year*) const

The number if days in the given month and year in the Gregorian calendar.

References isLeapYear(), and month().

Referenced by LCTime(), and test().

7.97.2.2 int UTIL::LCTime::daysInYear (int *year*) const

The number if days in the given year in the Gregorian calendar.

References isLeapYear().

7.97.2.3 std::string UTIL::LCTime::getDateString ( ) const

Date in human readable format, e.g.

10.02.2005 10:54:29.123456789 :

dd.mm.yyyy hh:mm:ss._ms_us_ns

Referenced by UTIL::LCTOOLS::dumpEventDetailed(), LCTime(), and test().

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

- LCTime.h
- LCTime.cc

7.98 UTIL::LCTokenizer Class Reference

Helper class for string tokenization.

`#include <UTIL/BitField64.h>`

Public Member Functions

- **LCTokenizer** (std::vector< std::string > &tokens, char del)
  Only c'tor, give (empty) token vector and delimeter character.
- void **operator()** (const char &c)
  Operator for use with algorithms, e.g.

7.98.1 Detailed Description

Helper class for string tokenization.

Usage:

std::vector<std::string> tokens ;

LCTokenizer (p. 162) t(tokens ,',') ;

std::for_each(aString.begin(), aString.end(), t) ;

7.98.2 Member Function Documentation

7.98.2.1 void UTIL::LCTokenizer::operator() (const char & *c*) [inline]

Operator for use with algorithms, e.g.

for_each

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

- BitField64.h

7.99 UTIL::LCTOOLS Class Reference

Provides some basic functions for printing event data.

`#include <UTIL/LCTOOLS.h>`

Static Public Member Functions

- static void **dumpEvent** (const **EVENT::LCEvent** ∗evt)
  Simple function to dump event contents to the screen.
- static void **dumpEventDetailed** (const **EVENT::LCEvent** ∗evt)

Dumps the whole event to the screen, using the dedicated print functions for the different collection types.

- static void **dumpRunHeader** (const **EVENT::LCRunHeader** ∗run)
  Simple function to dump the run header to the screen.
- static void **printMCParticles** (const **EVENT::LCCollection** ∗col)
  Complete printout of all MCParticles in the collection.
- static void **printSimTrackerHits** (const **EVENT::LCCollection** ∗col)
  Complete printout of all SimTrackerHits in the collection.
- static void **printTPCHits** (const **EVENT::LCCollection** ∗col)
  Complete printout of all TPCHits in the collection.
- static void **printTrackerRawData** (const **EVENT::LCCollection** ∗col)
  Complete printout of all TrackerRawData objects in the collection.
- static void **printTrackerData** (const **EVENT::LCCollection** ∗col)
  Complete printout of all TrackerData objects in the collection.
- static void **printTrackerPulse** (const **EVENT::LCCollection** ∗col)
  Complete printout of all TrackerPulse objects in the collection.
- static void **printTrackerHits** (const **EVENT::LCCollection** ∗col)
  Complete printout of all TrackerHits in the collection.
- static void **printTrackerHitPlane** (const **EVENT::LCCollection** ∗col)
  Complete printout of all TrackerHits in the collection.
- static void **printTrackerHitZCylinder** (const **EVENT::LCCollection** ∗col)
  Complete printout of all TrackerHits in the collection.
- static void **printSimCalorimeterHits** (const **EVENT::LCCollection** ∗col)
  Complete printout of all SimCalorimeterHits in the collection.
- static void **printCalorimeterHits** (const **EVENT::LCCollection** ∗col)
  Complete printout of all CalorimeterHits in the collection.
- static void **printRawCalorimeterHits** (const **EVENT::LCCollection** ∗col)
  Complete printout of all RawCalorimeterHits in the collection.
- static void **printLCFloatVecs** (const **EVENT::LCCollection** ∗col)
  Print LCFloatVec user extension vectors in the collection.
- static void **printLCIntVecs** (const **EVENT::LCCollection** ∗col)
  Print LCIntVecs user extension vectors in the collection.
- static void **printLCStrVecs** (const **EVENT::LCCollection** ∗col)
  Print LCStrVecs user extension vectors in the collection.
- static void **printTracks** (const **EVENT::LCCollection** ∗col)
  Complete printout of all Tracks in the collection.
- static void **printClusters** (const **EVENT::LCCollection** ∗col)
  Complete printout of all Clusters in the collection.
- static void **printVertices** (const **EVENT::LCCollection** ∗col)
  Complete printout of all Vertices in the collection.
- static void **printReconstructedParticles** (const **EVENT::LCCollection** ∗col)
  Complete printout of all ReconstructedParticles in the collection.
- static void **printLCGenericObjects** (const **EVENT::LCCollection** ∗col)
  Complete printout of all LCGenericObjects in the collection.
- static void **printRelation** (const **EVENT::LCCollection** ∗col)
  Print the relation.
- static void **printParameters** (const **EVENT::LCParameters** &params)
  Print the parameters.
- static std::string **getSimulatorStatusString** (const **EVENT::MCParticle** ∗mcp=0)
  Returns a string describing the simulator flag bits set for this particle.
- static int **printDaughterParticles** (const **EVENT::MCParticle** ∗part, int index)

7.99.1 Detailed Description

Provides some basic functions for printing event data.

Author

gaede

Version

Mar 10, 2003

See Also

LCEvent

7.99.2 Member Function Documentation

7.99.2.1 void UTIL::LCTOOLS::dumpEvent (const EVENT::LCEvent ∗ *evt*) `[static]`

Simple function to dump event contents to the screen.

Also demonstrates the access to event data through the **EVENT** (p. 21) interface. This is only a short format - one line per collection.

References EVENT::LCEvent::getCollection(), EVENT::LCEvent::getCollectionNames(), EVENT::LCEvent::getDetectorName(), EVENT::LCEvent::getEventNumber(), EVENT::LCCollection::getNumberOfElements(), EVENT::LCEvent::getRunNumber(), and EVENT::LCCollection::getTypeName().

7.99.2.2 std::string UTIL::LCTOOLS::getSimulatorStatusString (const EVENT::MCParticle ∗ *mcp* = 0) `[static]`

Returns a string describing the simulator flag bits set for this particle.

If no MCParticle given a string describing the bits is returned:

References EVENT::MCParticle::getSimulatorStatus(), EVENT::MCParticle::hasLeftDetector(), EVENT::MCParticle::isBackscatter(), EVENT::MCParticle::isCreatedInSimulation(), EVENT::MCParticle::isDecayedInCalorimeter(), EVENT::MCParticle::isDecayedInTracker(), EVENT::MCParticle::isOverlay(), EVENT::MCParticle::isStopped(), and EVENT::MCParticle::vertexIsNotEndpointOfParent().

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

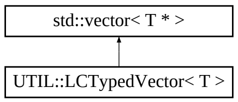
- UTIL/LCTOOLS.h
- LCTOOLS.cc

7.100 UTIL::LCTypedVector< T > Class Template Reference

Templated helper class that is an std::vector<T∗> of the elements in the LCCollection.

`#include <UTIL/LCTypedVector.h>`

Inheritance diagram for UTIL::LCTypedVector< T >:



Public Member Functions

- **LCTypedVector** (**EVENT::LCCollection** ∗col)
- **EVENT::LCCollection** ∗ **col** ()

Protected Attributes

- **EVENT::LCCollection** ∗ **_col**

7.100.1 Detailed Description

template<class T>class UTIL::LCTypedVector< T >

Templated helper class that is an std::vector<T∗> of the elements in the LCCollection.

This can be used, e.g. if the objects are to be passed to an algorithm in terms of iterators. For example:

LCTypedVector<SimTrackerHits> sim(evt->getCollection("tpc")) ;

LCCollectionVec∗ trackerHits = new LCCollectionVec(LCIO::TRACKERHIT) ;

std::transform(sim.begin(), sim.end(), std::back_inserter(∗trackerHits) , digitizer) ;

Note that instantiating an **LCTypedVector** (p. 164) involves copying and casting of the pointers in the collection. In a future release of LCIO this might not be neccessary any more.

Author

gaede

Version

Id:

LCTypedVector.h (p. **??**),v 1.3 2007-02-19 17:38:46 gaede Exp

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

- LCTypedVector.h

7.101 UTIL::LCWarning Class Reference

Utility class to show warnings in LCIO.

```
#include <UTIL/LCWarning.h>
```

Public Member Functions

- void **registerWarning** (const std::string &id, const std::string &txt, int max=10)
  register a new warning
- void **printWarning** (const std::string &id)
  print the warning text associated to the given id

Static Public Member Functions

- static **LCWarning** & **getInstance** ()

7.101.1 Detailed Description

Utility class to show warnings in LCIO.

The maximum amount of warnings printed is configurable and per default set to 10.

Warnings are shown one last time when application ends.

Author

engels

Version

7.101.2 Member Function Documentation

7.101.2.1 void UTIL::LCWarning::registerWarning (const std::string & *id*, const std::string & *txt*, int *max* = `10`)

register a new warning

id = "a_unique_string_to_identify_the_warning" txt = "the world will end tomorrow, leave the room immediately and go get some fun ;)" max = the maximum amount of times to display the warning

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

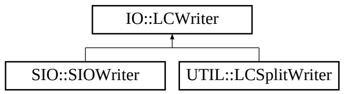
- LCWarning.h
- LCWarning.cc

7.102 IO::LCWriter Class Reference

Interface for writing data with LCIO.

```
#include <pre-generated/IO/LCWriter.h>
```

Inheritance diagram for IO::LCWriter:



Public Member Functions

- virtual ∼**LCWriter** ()
  Destructor.
- virtual void **open** (const std::string &filename)=0 throw (IOException, std::exception)
  Opens a file for writing.
- virtual void **open** (const std::string &filename, int writeMode)=0 throw (IOException, std::exception)
  Opens a file for writing.
- virtual void **setCompressionLevel** (int level)=0
  *Set the compression level - needs to be called before **open()** (p. 167) otherwise call will have no effect.*
- virtual void **writeRunHeader** (const **EVENT::LCRunHeader** ∗hdr)=0 throw (IOException, std::exception)
  Writes the given run header to file.

- virtual void **writeEvent** (const **EVENT::LCEvent** ∗evt)=0 throw (IOException, std::exception)
  Writes the given event to file.
- virtual void **close** ()=0 throw (IOException, std::exception)
  Closes the output file/stream.
- virtual void **flush** ()=0 throw (IOException, std::exception)
  Flushes the output file/stream.

7.102.1 Detailed Description

Interface for writing data with LCIO.

Uses interfaces from EVENT/hep.lcio.event. Use LCFactory to instantiate a corresponding **LCWriter** (p. 166) object for the output format at hand (**SIO** (p. 30) only, so far).

See Also

LCFactory

Author

gaede

Version

Mar 4, 2003

7.102.2 Member Function Documentation

7.102.2.1 virtual void IO::LCWriter::close ( ) throw **IOException**, std::exception) [pure virtual]

Closes the output file/stream.

Exceptions

IOException *(p. 79)*	

Implemented in **SIO::SIOWriter** (p. 255), and **UTIL::LCSplitWriter** (p. 153).

7.102.2.2 virtual void IO::LCWriter::flush ( ) throw **IOException**, std::exception) [pure virtual]

Flushes the output file/stream.

Exceptions

IOException *(p. 79)*	

Implemented in **SIO::SIOWriter** (p. 256), and **UTIL::LCSplitWriter** (p. 153).

7.102.2.3 virtual void IO::LCWriter::open (const std::string & *filename*) throw **IOException**, std::exception) [pure virtual]

Opens a file for writing.

If file with given name exists, an exception is thrown. Use append or new mode instead.

Exceptions

IOException *(p. 79)*	

Implemented in **SIO::SIOWriter** (p. 256), and **UTIL::LCSplitWriter** (p. 153).

7.102.2.4 virtual void IO::LCWriter::open (const std::string & *filename*, int *writeMode*) throw **IOException**, std::exception) [pure virtual]

Opens a file for writing.

Possible write modes are: LCIO::WRITE_NEW (existing files are replaced) and LCIO::WRITE_APPEND.

Exceptions

IOException *(p. 79)*	

Implemented in **SIO::SIOWriter** (p. 256), and **UTIL::LCSplitWriter** (p. 154).

7.102.2.5 virtual void IO::LCWriter::setCompressionLevel (int *level*) [pure virtual]

Set the compression level - needs to be called before **open()** (p. 167) otherwise call will have no effect.

If not called the Writer will use default compression.

Valid compression levels are:

- level $<$ 0 : default compression
- level == 0 : no compression
- level $>$ 0 : compression level (typically 1 (fastest) - 9 (best compression))

Status: (v01-09)

C++: experimental code - don't use for production

Java: not implemented

Parameters

level	compression level

Implemented in **SIO::SIOWriter** (p. 256), and **UTIL::LCSplitWriter** (p. 154).

Referenced by UTIL::LCSplitWriter::setCompressionLevel().

7.102.2.6 virtual void IO::LCWriter::writeEvent (const **EVENT::LCEvent** ∗ *evt*) throw **IOException**, std::exception) [pure virtual]

Writes the given event to file.

Exceptions

IOException *(p. 79)*	

Implemented in **SIO::SIOWriter** (p. 257), and **UTIL::LCSplitWriter** (p. 154).

7.102.2.7 virtual void IO::LCWriter::writeRunHeader (const **EVENT::LCRunHeader** ∗ *hdr*) throw **IOException**, std::exception) [pure virtual]

Writes the given run header to file.

Exceptions

IOException *(p. 79)*	

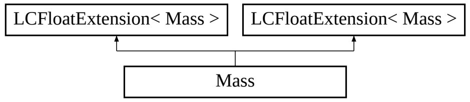
Implemented in **SIO::SIOWriter** (p. 257), and **UTIL::LCSplitWriter** (p. 154).

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

- LCWriter.h

7.103 Mass Struct Reference

Inheritance diagram for Mass:



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

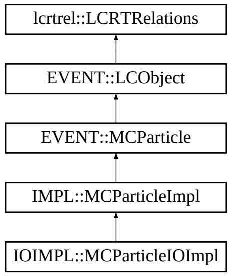
- lcrtrelation.cc

7.104 EVENT::MCParticle Class Reference

The **LCIO** (p. 115) Monte Carlo particle.

```
#include <pre-generated/EVENT/MCParticle.h>
```

Inheritance diagram for EVENT::MCParticle:



Public Types

- typedef **MCParticle lcobject_type**

  *Useful typedef for template programming with **LCIO** (p. 115).*

Public Member Functions

- virtual ∼**MCParticle** ()

  Destructor.
- virtual double **getEnergy** () const =0

  Returns the energy of the particle (at the vertex) in [GeV] computed from the particle's momentum and mass - only float used in files.
- virtual const float ∗ **getSpin** () const =0

  Returns the spin (helicity) vector of the particle.
- virtual const int ∗ **getColorFlow** () const =0

  Returns the color flow as defined by the generator.
- virtual const **MCParticleVec** & **getParents** () const =0

  Returns the parents of this particle.
- virtual const **MCParticleVec** & **getDaughters** () const =0

  Returns the daughters of this particle.
- virtual int **getPDG** () const =0

  Returns the PDG code of the particle.
- virtual int **getGeneratorStatus** () const =0

  Returns the status for particles as defined by the generator, typically
  0 empty line
  1 undecayed particle, stable in the generator
  2 particle decayed in the generator
  3 documentation line.
- virtual int **getSimulatorStatus** () const =0

  Returns the status for particles from the simulation, e.g.
- virtual bool **isCreatedInSimulation** () const =0

  True if the particle has been created by the simulation program (rather than the generator).
- virtual bool **isBackscatter** () const =0

  True if the particle was created by the simulator as a result of an interaction or decay in non-tracking region, e.g.
- virtual bool **vertexIsNotEndpointOfParent** () const =0

  True if the particle was created as a result of a continuous process where the parent particle continues, i.e.
- virtual bool **isDecayedInTracker** () const =0

  True if the particle decayed or interacted in a tracking region.
- virtual bool **isDecayedInCalorimeter** () const =0

  True if the particle decayed or interacted (non-continuous interaction, particle terminated) in non-tracking region.
- virtual bool **hasLeftDetector** () const =0

  True if the particle left the world volume undecayed.
- virtual bool **isStopped** () const =0

  True if the particle lost all kinetic energy inside the world volume and did not decay.
- virtual bool **isOverlay** () const =0

  True if the particle has been overlayed by the simulation (or digitization) program.
- virtual const double ∗ **getVertex** () const =0

  Returns the production vertex of the particle in [mm].
- virtual float **getTime** () const =0

  The creation time of the particle in [ns] wrt.
- virtual const double ∗ **getEndpoint** () const =0

  Returns the endpoint of the particle in [mm] if the endpoint has been set explicetly.
- virtual const double ∗ **getMomentum** () const =0

  Returns the particle's 3-momentum at the production vertex in [GeV].
- virtual const double ∗ **getMomentumAtEndpoint** () const =0

  Returns the particle's 3-momentum at the endpoint in [GeV] Only float used in files.
- virtual double **getMass** () const =0

  Returns the mass of the particle in [GeV] - only float used in files.
- virtual float **getCharge** () const =0

  Returns the particle's charge.

Static Public Attributes

- static const int **BITEndpoint** = 31
- static const int **BITCreatedInSimulation** = 30
- static const int **BITBackscatter** = 29
- static const int **BITVertexIsNotEndpointOfParent** = 28
- static const int **BITDecayedInTracker** = 27
- static const int **BITDecayedInCalorimeter** = 26
- static const int **BITLeftDetector** = 25
- static const int **BITStopped** = 24
- static const int **BITOverlay** = 23

Additional Inherited Members

7.104.1 Detailed Description

The **LCIO** (p. 115) Monte Carlo particle.

Author

gaede

Version

Id:

MCParticle.aid,v 1.26 2006-08-03 16:53:34 gaede Exp

7.104.2 Member Function Documentation

7.104.2.1 virtual const double* EVENT::MCParticle::getEndpoint ( ) const [pure virtual]

Returns the endpoint of the particle in [mm] if the endpoint has been set explicetly.

If not, the vertex of the first daughter that has vertexIsNotEndpointOfParent==false is returned. If no such daughter exists (0,0,0) is returned.

Implemented in **IMPL::MCParticleImpl** (p. 176).

Referenced by UTIL::LCTOOLS::printMCParticles(), and SIO::SIOParticleHandler::write().

7.104.2.2 virtual const double* EVENT::MCParticle::getMomentum ( ) const [pure virtual]

Returns the particle's 3-momentum at the production vertex in [GeV].

- only float used in files.

Implemented in **IMPL::MCParticleImpl** (p. 174).

Referenced by UTIL::LCTOOLS::printMCParticles(), and SIO::SIOParticleHandler::write().

7.104.2.3 virtual int EVENT::MCParticle::getSimulatorStatus ( ) const [pure virtual]

Returns the status for particles from the simulation, e.g.

decayed in flight. Bits 31-16 are used to decode the information. Use the followng boolean functions to determine the proper simulator status:

See Also

isCreatedInSimulation() (p. 170)
isBackscatter() (p. 172)
vertexIsNotEndpointOfParent() (p. 172)
isDecayedInTracker() (p. 170)
isDecayedInCalorimeter() (p. 170)
hasLeftDetector() (p. 170)
isStopped() (p. 170)
isOverlay() (p. 170)

Implemented in **IMPL::MCParticleImpl** (p. 176).

Referenced by UTIL::LCTOOLS::getSimulatorStatusString(), and SIO::SIOParticleHandler::write().

7.104.2.4 virtual float EVENT::MCParticle::getTime ( ) const [pure virtual]

The creation time of the particle in [ns] wrt.

the event, e.g. for preassigned decays or decays in flight from the simulator.

Implemented in **IMPL::MCParticleImpl** (p. 177).

Referenced by SIO::SIOParticleHandler::write().

7.104.2.5 virtual bool EVENT::MCParticle::isBackscatter ( ) const [pure virtual]

True if the particle was created by the simulator as a result of an interaction or decay in non-tracking region, e.g.

a calorimeter. By convention, such particles are not saved. However, if this particle creates a tracker hit, the particle is added to the **MCParticle** (p. 169) list with this flag set, and the parent set to the particle that initially decayed or interacted in a non-tracking region.

Implemented in **IMPL::MCParticleImpl** (p. 174).

Referenced by UTIL::LCTOOLS::getSimulatorStatusString().

7.104.2.6 virtual bool EVENT::MCParticle::vertexIsNotEndpointOfParent ( ) const [pure virtual]

True if the particle was created as a result of a continuous process where the parent particle continues, i.e.

hard ionization, Bremsstrahlung, elastic interactions, etc.

Implemented in **IMPL::MCParticleImpl** (p. 174).

Referenced by UTIL::LCTOOLS::getSimulatorStatusString().

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

- MCParticle.h
- MCParticleImpl.cc

7.105 IMPL::MCParticleCont Struct Reference

Public Member Functions

- **MCParticleCont** (**EVENT::MCParticle** ∗part, float e, float t, int pdg, const float ∗step)

Public Attributes

- **EVENT::MCParticle** ∗ **Particle**
- float **Energy**
- float **Time**
- int **PDG**
- float **StepPosition** [3]

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

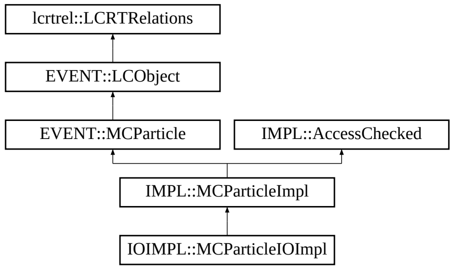
- SimCalorimeterHitImpl.h

7.106 IMPL::MCParticleImpl Class Reference

Implementation of MCParticle.

`#include <IMPL/MCParticleImpl.h>`

Inheritance diagram for IMPL::MCParticleImpl:



Public Member Functions

- virtual ∼**MCParticleImpl** ()
  Copy c'tor.
- virtual int **id** () const
  Returns an object id for internal (debugging) use in LCIO.
- virtual const **EVENT::MCParticleVec** & **getParents** () const
  Returns the parents of this particle.
- virtual const **EVENT::MCParticleVec** & **getDaughters** () const
  Returns the daughters of this particle.
- virtual const double ∗ **getEndpoint** () const
  Returns the i-th daughter of this particle.
- virtual int **getPDG** () const
  Returns the number of daughters of this particle.

- virtual int **getGeneratorStatus** () const
  Returns the status for particles from the generator
  0 empty line
  1 undecayed particle, stable in the generator
  2 particle decayed in the generator
  3 documentation line.
- virtual int **getSimulatorStatus** () const
  Returns the status for particles from the simulation, e.g.
- virtual bool **isCreatedInSimulation** () const
  True if the particle has been created by the simulation program (rather than the generator).
- virtual bool **isBackscatter** () const
  True if the particle is the result of a backscatter from a calorimeter shower.
- virtual bool **vertexIsNotEndpointOfParent** () const
  True if the particle's vertex is not the endpoint of the parent particle.
- virtual bool **isDecayedInTracker** () const
  True if the particle has interacted in a tracking region.
- virtual bool **isDecayedInCalorimeter** () const
  True if the particle has interacted in a calorimeter region.
- virtual bool **hasLeftDetector** () const
  True if the particle has left the world volume undecayed.
- virtual bool **isStopped** () const
  True if the particle has been stopped by the simulation program.
- virtual bool **isOverlay** () const
  True if the particle has been overlayed by the simulation (or digitization) program.
- virtual const double ∗ **getVertex** () const
  Returns the production vertex of the particle.
- virtual float **getTime** () const
  The creation time of the particle in [ns] wrt.
- virtual const double ∗ **getMomentum** () const
  Returns the particle momentum at the production vertex.
- virtual const double ∗ **getMomentumAtEndpoint** () const
  Returns the particle momentum at the endpoint.
- virtual double **getMass** () const
  Returns the mass of the particle in [GeV].
- virtual float **getCharge** () const
  Returns the particle's charge.
- virtual double **getEnergy** () const
  Returns the energy of the particle (at the vertex) in [GeV] computed from the particle's momentum and mass.
- virtual const float ∗ **getSpin** () const
  Returns the spin.
- virtual const int ∗ **getColorFlow** () const
  Returns the color flow.
- void **addParent** (**EVENT::MCParticle** ∗mom)
  Adds a parent particle.
- void **setPDG** (int pdg)
  Sets the parent.
- void **setGeneratorStatus** (int status)
  Sets the Generator status.
- void **setSimulatorStatus** (int status)
  Sets the Simulator status.
- void **setVertex** (const double vtx[3])

Sets the production vertex.
- void **setTime** (float time)
  Sets the createion time.
- void **setEndpoint** (const double pnt[3])
  Sets the particle endpoint.
- void **setMomentum** (const float p[3])
  Sets the momentum.
- void **setMomentum** (const double p[3])
  Sets the momentum.
- void **setMomentumAtEndpoint** (const float p[3])
  Sets the momentum at the endpoint.
- void **setMomentumAtEndpoint** (const double p[3])
  Sets the momentum at the endpoint.
- void **setMass** (float m)
  Sets the mass.
- void **setCharge** (float c)
  Sets the charge.
- void **setSpin** (const float spin[3])
  Sets the spin.
- void **setColorFlow** (const int cflow[2])
  Sets the color flow.
- virtual void **setCreatedInSimulation** (bool val)
- virtual void **setBackscatter** (bool val)
- virtual void **setVertexIsNotEndpointOfParent** (bool val)
- virtual void **setDecayedInTracker** (bool val)
- virtual void **setDecayedInCalorimeter** (bool val)
- virtual void **setHasLeftDetector** (bool val)
- virtual void **setStopped** (bool val)
- virtual void **setOverlay** (bool val)

Protected Member Functions

- void **addDaughter** (**EVENT::MCParticle** ∗daughter)
  Adds a daughter particle - only called from ***addParent()*** *(p. 174).*

Protected Attributes

- int **_pdg**
- int **_genstatus**
- std::bitset< 32 > **_simstatus**
- double **_vertex** [3]
- double **_endpoint** [3]
- double **_pEndpoint** [3]
- double **_p** [3]
- double **_mass**
- float **_charge**
- float **_time**
- **EVENT::MCParticleVec _parents**
- **EVENT::MCParticleVec _daughters**
- bool **_endpointSet**
- float **_spin** [3]
- int **_colorFlow** [2]

Additional Inherited Members

7.106.1 Detailed Description

Implementation of MCParticle.

Author

gaede

Version

Id:

MCParticleImpl.h (p. **??**),v 1.24 2010-01-21 16:28:25 gaede Exp

7.106.2 Constructor & Destructor Documentation

7.106.2.1 IMPL::MCParticleImpl::∼MCParticleImpl ( ) [virtual]

Copy c'tor.

Destructor.

7.106.3 Member Function Documentation

7.106.3.1 const double ∗ IMPL::MCParticleImpl::getEndpoint ( ) const [virtual]

Returns the i-th daughter of this particle.

Same as getDaughter() except for return type.

See Also

getNumberOfDaughtersReturns the number of parents of this particle - 0 if mother.Returns the i-th parent of this particle.Returns the endpoint of the particle in [mm] - returns NULL if undefined (not set). Definition of the enpoint depends on the application that created the particle, e.g. the start point of the shower in a calorimeter.

Implements **EVENT::MCParticle** (p. 171).

7.106.3.2 int IMPL::MCParticleImpl::getPDG ( ) const [virtual]

Returns the number of daughters of this particle.

Returns the PDG code of the particle.

Implements **EVENT::MCParticle** (p. 170).

Referenced by UTIL::LCStdHepRdr::readEvent(), and UTIL::LCStdHepRdrNew::readEvent().

7.106.3.3 int IMPL::MCParticleImpl::getSimulatorStatus ( ) const [virtual]

Returns the status for particles from the simulation, e.g.

decayed in flight. Bits 31-16 are used to decode the information. Use the followng boolean functions to determine the proper simulator status:

See Also

isCreatedInSimulation() (p. 174)
isBackscatter() (p. 174)
vertexIsNotEndpointOfParent() (p. 174)
isDecayedInTracker() (p. 174)
isDecayedInCalorimeter() (p. 174)
hasLeftDetector() (p. 174)
isStopped() (p. 174)
isOverlay() (p. 174)

Implements **EVENT::MCParticle** (p. 172).

7.106.3.4 float IMPL::MCParticleImpl::getTime ( ) const `[virtual]`

The creation time of the particle in [ns] wrt.

the event, e.g. for preassigned decays or decays in flight from the simulator.

Implements **EVENT::MCParticle** (p. 172).

7.106.3.5 void IMPL::MCParticleImpl::setPDG (int *pdg*)

Sets the parent.

Sets a second parent.Sets the PDG code.

Referenced by HEPEVTIMPL::HEPEVT::fromHepEvt(), UTIL::LCStdHepRdrNew::readEvent(), and UTIL::LCStdHepRdr::readEvent().

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

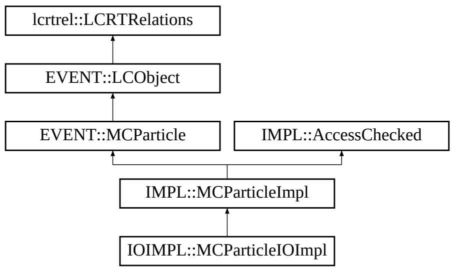
- MCParticleImpl.h
- MCParticleImpl.cc

7.107 IOIMPL::MCParticleIOImpl Class Reference

Adding stuff needed for io (friend declarations, etc.)

`#include <IOIMPL/MCParticleIOImpl.h>`

Inheritance diagram for IOIMPL::MCParticleIOImpl:



Friends

- class **SIO::SIOParticleHandler**

Additional Inherited Members

7.107.1 Detailed Description

Adding stuff needed for io (friend declarations, etc.)

Author

gaede

Version

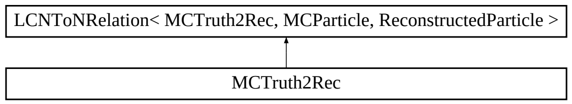
Mar 7, 2003

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

- MCParticleIOImpl.h

7.108 MCTruth2Rec Struct Reference

Inheritance diagram for MCTruth2Rec:

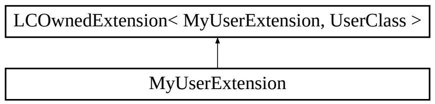


The documentation for this struct was generated from the following file:

- test_RTextension.cc

7.109 MyUserExtension Struct Reference

Inheritance diagram for MyUserExtension:



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

- lcrtrelation.cc

7.110 lcrtrel_helper::NoDelete Struct Reference

Empty delete function for pointers w/o ownership.

`#include <LCRTRelations.h>`

Static Public Member Functions

- static void **clean** (void $*$)

7.110.1 Detailed Description

Empty delete function for pointers w/o ownership.

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

- LCRTRelations.h

7.111 lcrtrel_helper::objorcont< is_container > Struct Template Reference

Helper functions that treat single objects and containers.

`#include <LCRTRelations.h>`

Static Public Member Functions

- template<class T , class S >
  static void **add** (T t, S s)
- template<class T , class S >
  static void **remove** (T t, S s)

7.111.1 Detailed Description

template<bool is_container>struct lcrtrel_helper::objorcont< is_container >

Helper functions that treat single objects and containers.

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

- LCRTRelations.h

7.112 lcrtrel_helper::objorcont< false > Struct Template Reference

Helper functions specialization for single objects.

`#include <LCRTRelations.h>`

Static Public Member Functions

- template<class T , class S >
  static void **add** (T &t, S s)
- template<class T , class S >
  static void **remove** (T &t, S)

7.112.1 Detailed Description

template<>struct lcrtrel_helper::objorcont< false >

Helper functions specialization for single objects.

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

- LCRTRelations.h

7.113 opts_t Struct Reference

Public Attributes

- const char $*$ **colName**
- char $**$ **inputFiles**
- int **inputFilesNum**
- int **verbosity**
- bool **pedantic**
- bool **average**
- int **startevent**
- int **maxevents**
- int **expelements**
- int **minelements**
- int **maxelements**
- int **abselementerror**
- int **abseventerror**
- float **relelementerror**
- float **releventerror**
- int **elementsTotal**
- int **eventsTotal**
- int **eventsFailed**
- int **eventsSkipped**

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

- lcio_check_col_elements.cc

7.114 pagerInfo Struct Reference

Public Attributes

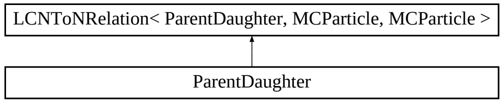
- char $*$ **filename**
- int **fd_temp**
- int **fd_old**
- bool **save**

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

- lsh.cc

7.115 ParentDaughter Struct Reference

Inheritance diagram for ParentDaughter:



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

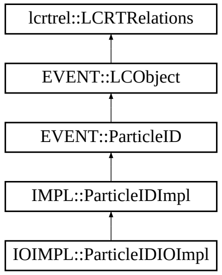
- lcrtrelation.cc

7.116 EVENT::ParticleID Class Reference

Persistent interface for **LCIO** (p. 115) **ParticleIDs** (p. 185).

`#include <pre-generated/EVENT/ParticleID.h>`

Inheritance diagram for EVENT::ParticleID:



Public Types

- typedef **ParticleID lcobject_type**

  Useful typedef for template programming with ***LCIO*** *(p. 115).*

Public Member Functions

- virtual ∼**ParticleID** ()

  Destructor.
- virtual int **getType** () const =0

  Type - userdefined.
- virtual int **getPDG** () const =0

  The PDG code of this id - UnknownPDG (999999) if unknown.
- virtual float **getLikelihood** () const =0

  The likelihood of this hypothesis - in a user defined normalization.
- virtual int **getAlgorithmType** () const =0

  Type of the algorithm/module that created this hypothesis - NOTE: must be unique within one collection.

- virtual const **FloatVec** & **getParameters** () const =0

  Parameters associated with this hypothesis.

Static Public Attributes

- static const int **UnknownPDG** = 999999

  Constant to be used if the PDG code is not known or undefined.

Additional Inherited Members

7.116.1 Detailed Description

Persistent interface for **LCIO** (p. 115) **ParticleIDs** (p. 185).

Used by **ReconstructedParticle** (p. 193) and **Cluster** (p. 58) for different hypotheses on the particle type.

Author

gaede

Version

Id:

ParticleID.aid,v 1.11 2008-05-30 13:23:27 gaede Exp

See Also

ReconstructedParticle.getParticleIDs() (p. 196)
Cluster.getParticleIDs() (p. 60)

7.116.2 Member Function Documentation

7.116.2.1 virtual int EVENT::ParticleID::getAlgorithmType ( ) const [pure virtual]

Type of the algorithm/module that created this hypothesis - NOTE: must be unique within one collection.

Check/set collection parameters PIDAlgorithmTypeName and PIDAlgorithmTypeID.

Implemented in **IMPL::ParticleIDImpl** (p. 184).

Referenced by UTIL::LCTOOLS::printClusters(), UTIL::LCTOOLS::printReconstructedParticles(), SIO::SIOClusterHandler::write(), and SIO::SIOReconstructedParticleHandler::write().

7.116.2.2 virtual const FloatVec& EVENT::ParticleID::getParameters ( ) const [pure virtual]

Parameters associated with this hypothesis.

Check/set collection parameters ParameterNames_PIDAlgorithmTypeName for decoding the indices.

Implemented in **IMPL::ParticleIDImpl** (p. 184).

Referenced by UTIL::LCTOOLS::printClusters(), UTIL::LCTOOLS::printReconstructedParticles(), SIO::SIOClusterHandler::write(), and SIO::SIOReconstructedParticleHandler::write().

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

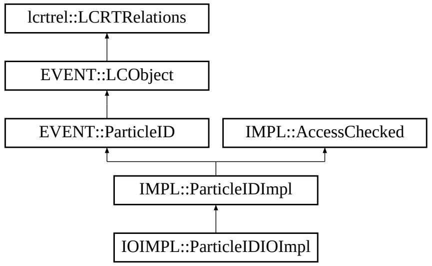
- ParticleID.h

7.117 IMPL::ParticleIDImpl Class Reference

Implementation of ParticleID.

`#include <IMPL/ParticleIDImpl.h>`

Inheritance diagram for IMPL::ParticleIDImpl:



Public Member Functions

- **ParticleIDImpl** ()
  Default constructor, initializes values to 0.
- virtual ∼**ParticleIDImpl** ()
  Destructor.
- virtual int **id** () const
  Returns an object id for internal (debugging) use in LCIO.
- virtual int **getType** () const
  Type - userdefined.
- virtual int **getPDG** () const
  The PDG code of this id - UnknownPDG (999999) if unknown.
- virtual float **getLikelihood** () const
  The likelihood of this hypothesis - in a user defined normalization.
- virtual int **getAlgorithmType** () const
  Type of the algorithm/module that created this hypothesis - NOTE: must be unique within one collection.
- virtual const **EVENT::FloatVec** & **getParameters** () const
  Parameters associated with this hypothesis.
- virtual **EVENT::FloatVec** & **parameters** ()
  Access to parameters associated with this hypothesis.
- void **setType** (int type)
- void **setPDG** (int pdg)
- void **setLikelihood** (float logL)
- void **setAlgorithmType** (int algorithmType)
- void **addParameter** (float p)

Protected Attributes

- int **_type**
- int **_pdg**
- float **_likelihood**
- int **_algorithmType**
- **EVENT::FloatVec _parameters**

Additional Inherited Members

7.117.1 Detailed Description

Implementation of ParticleID.

See Also

ParticleID

Author

gaede

Version

Id:

ParticleIDImpl.h (p. **??**),v 1.11 2008-05-30 13:23:27 gaede Exp

7.117.2 Member Function Documentation

7.117.2.1 int IMPL::ParticleIDImpl::getAlgorithmType ( ) const [virtual]

Type of the algorithm/module that created this hypothesis - NOTE: must be unique within one collection.

Check/set collection parameters PIDAlgorithmTypeName and PIDAlgorithmTypeID.

Implements **EVENT::ParticleID** (p. 182).

7.117.2.2 const EVENT::FloatVec & IMPL::ParticleIDImpl::getParameters ( ) const [virtual]

Parameters associated with this hypothesis.

Check/set collection parameter ParameterNames_PIDAlgorithmTypeName for decoding the indices.

Implements **EVENT::ParticleID** (p. 182).

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

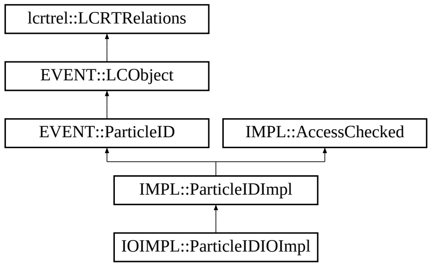
- ParticleIDImpl.h
- ParticleIDImpl.cc

7.118 IOIMPL::ParticleIDIOImpl Class Reference

Adding stuff needed for io (friend declarations, etc.)

`#include <IOIMPL/ParticleIDIOImpl.h>`

Inheritance diagram for IOIMPL::ParticleIDIOImpl:



Friends

- class **SIO::SIOReconstructedParticleHandler**
- class **SIO::SIOClusterHandler**

Additional Inherited Members

7.118.1 Detailed Description

Adding stuff needed for io (friend declarations, etc.)

Author

gaede

Version

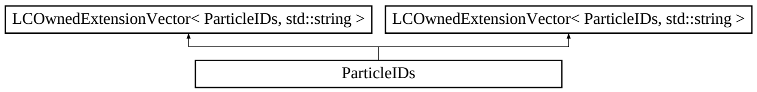
Mar 31, 2004

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

- ParticleIDIOImpl.h

7.119 ParticleIDs Struct Reference

Inheritance diagram for ParticleIDs:



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

- lcrtrelation.cc

7.120 UTIL::PIDHandler Class Reference

Convenient class for setting and retrieving particle id information attached to ReconstructedParticles.

`#include <UTIL/PIDHandler.h>`

Public Member Functions

- **PIDHandler** (const **LCCollection** ∗col)

  Create ***PIDHandler*** *(p. 186) for reading from the given collection - read the collection parameters PIDAlgorithmType-Name, PIDAlgorithmTypeID and the corresponding ParameterNames_PIDAlgorithmTypeName if they exist.*
- **PIDHandler** (**LCCollection** ∗col)

  Create ***PIDHandler*** *(p. 186) for setting PID information in the given collection.*
- ∼**PIDHandler** ()

  Update the collection parameter when going out of scope.
- int **addAlgorithm** (const std::string &algoName, const **StringVec** &parameterNames)

  Add a new algorithm and return the corresponding algorithm id.
- int **getAlgorithmID** (const std::string &algoName)

  Return the typeID of algorithm algoName - throws ***UnknownAlgorithm*** *(p. 311).*
- const std::string & **getAlgorithmName** (int algoID)

  Return the name of the algorithm with id - throws ***UnknownAlgorithm*** *(p. 311).*
- int **getParameterIndex** (int algorithmID, const std::string &pName)

  The index of parameter pName for the algorithm with algorithmID - throws UnknownAlgoritm.
- const **ParticleID** & **getParticleID** (**LCObject** ∗particle, int algorithmID)

  Return the (first) ParticleID object for the given algorithm and particle (or cluster) - throws ***UnknownAlgorithm*** *(p. 311).*
- **ParticleIDVec getParticleIDs** (**LCObject** ∗p, int id)

  Return all PID objects for a given algorithm - ordered with decreasing likelihood - throws ***UnknownAlgorithm*** *(p. 311).*
- void **setParticleIDUsed** (**IMPL::ReconstructedParticleImpl** ∗particle, int algorithmID)

  Set the particleID algorithm that is used for this particle's kinematic variables.
- const **StringVec** & **getParameterNames** (int algorithmID)

  The names of parameters for the algorithm with algorithmID - throws UnknownAlgoritm.
- const **IntVec** & **getAlgorithmIDs** ()

  Return the IDs of all known Algorithms.
- void **setParticleID** (**LCObject** ∗p, int userType, int PDG, float likelihood, int algorithmID, const **FloatVec** &params)

  Set the particleID information for this particle (or cluster) - throws ***UnknownAlgorithm*** *(p. 311).*

Protected Member Functions

- int **nextID** ()
- void **init** (const **LCCollection** ∗col)

Protected Attributes

- **LCCollection** ∗ **_col**
- **CPM _cpm**
- int **_type**
- int **_maxID**
- PNM **_pNames**
- CPMINV **_cpmInv**
- **IntVec _ids**

7.120.1 Detailed Description

Convenient class for setting and retrieving particle id information attached to ReconstructedParticles.

Use one instance of **PIDHandler** (p. 186) for one LCCollection only. When used for setting particle id information the scope of the **PIDHandler** (p. 186) must match that of the collection, as the collection parameters are updated automatically when the **PIDHandler** (p. 186) object goes out of scope. Reads (and updates) the collection parameters PIDAlgorithmTypeName, PIDAlgorithmTypeID and the corresponding ParameterNames_PIDAlgorithmTypeName for every algorithm used.

See Also

ReconstructedParticle
ParticleID

Author

F.Gaede, DESY

Version

Id:

PIDHandler.h (p. **??**),v 1.2 2010-01-21 16:28:25 gaede Exp

7.120.2 Constructor & Destructor Documentation

7.120.2.1 UTIL::PIDHandler::PIDHandler (**LCCollection** ∗ *col*)

Create **PIDHandler** (p. 186) for setting PID information in the given collection.

- read and update the collection parameters PIDAlgorithmTypeName, PIDAlgorithmTypeID and the corresponding ParameterNames_PIDAlgorithmTypeName if they exist.

7.120.3 Member Function Documentation

7.120.3.1 const **ParticleID** & UTIL::PIDHandler::getParticleID (**LCObject** ∗ *particle,* int *algorithmID*)

Return the (first) ParticleID object for the given algorithm and particle (or cluster) - throws **UnknownAlgorithm** (p. 311).

Only use if you know there is only one PID object for the algorithms or if you simply want the most likely PID for this algorithm.

References getParticleIDs().

7.120.3.2 void UTIL::PIDHandler::setParticleIDUsed (**IMPL::ReconstructedParticleImpl** ∗ *particle,* int *algorithmID*)

Set the particleID algorithm that is used for this particle's kinematic variables.

- throws **UnknownAlgorithm** (p. 311).

References IMPL::ReconstructedParticleImpl::getParticleIDs().

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

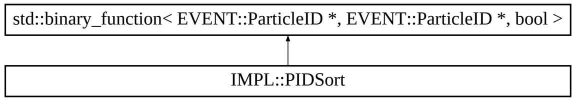
- PIDHandler.h
- PIDHandler.cc

7.121 IMPL::PIDSort Class Reference

Helper class to sort **ParticleIDs** (p. 185) wrt.

`#include <IMPL/ParticleIDImpl.h>`

Inheritance diagram for IMPL::PIDSort:



Public Member Functions

- bool **operator()** (const **EVENT::ParticleID** ∗p1, const **EVENT::ParticleID** ∗p2)

7.121.1 Detailed Description

Helper class to sort **ParticleIDs** (p. 185) wrt.

their likelihood.

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

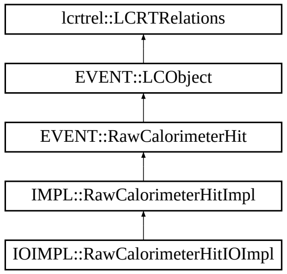
- ParticleIDImpl.h

7.122 EVENT::RawCalorimeterHit Class Reference

The generic calorimeter hit for real data (or simulation thereof).

`#include <pre-generated/EVENT/RawCalorimeterHit.h>`

Inheritance diagram for EVENT::RawCalorimeterHit:



Public Types

- typedef **RawCalorimeterHit lcobject_type**

  Useful typedef for template programming with ***LCIO*** *(p. 115).*

Public Member Functions

- virtual ∼**RawCalorimeterHit** ()
  Destructor.
- virtual int **getCellID0** () const =0
  Returns the detector specific (geometrical) cell id.
- virtual int **getCellID1** () const =0
  Returns the second detector specific (geometrical) cell id.
- virtual int **getAmplitude** () const =0
  Returns the amplitude of the hit in ADC counts.
- virtual int **getTimeStamp** () const =0
  Returns a time stamp for the hit.

Additional Inherited Members

7.122.1 Detailed Description

The generic calorimeter hit for real data (or simulation thereof).

In order to reduce the file size you can suppress a (32-bit) pointer tag stored with every hit by setting the flag(LCIO::RCHBIT_NO_PTR)==1.

NB: If you apply this flag to reduce the file size you won't be able to point/refer to the RawCalorimeterHits, e.g. from an LCRelation (p. 136) object.

Author

gaede

Version

Id:

RawCalorimeterHit.aid,v 1.4 2006-08-03 16:53:35 gaede Exp

7.122.2 Member Function Documentation

7.122.2.1 virtual int EVENT::RawCalorimeterHit::getCellID1 ( ) const [pure virtual]

Returns the second detector specific (geometrical) cell id.

Optional, check/set flag(LCIO::RCHBIT_ID1)==1.

Implemented in **IMPL::RawCalorimeterHitImpl** (p. 191).

Referenced by SIO::SIORawCalHitHandler::write().

7.122.2.2 virtual int EVENT::RawCalorimeterHit::getTimeStamp ( ) const [pure virtual]

Returns a time stamp for the hit.

Optional, check/set flag(LCIO::RCHBIT_TIME)==1.

Implemented in **IMPL::RawCalorimeterHitImpl** (p. 191).

Referenced by SIO::SIORawCalHitHandler::write().

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

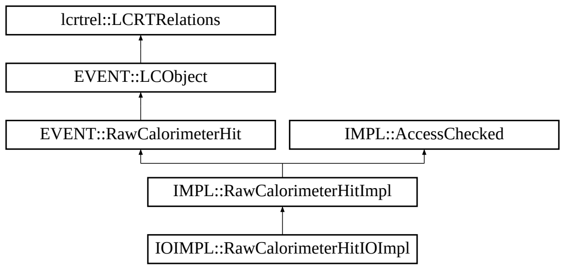
- RawCalorimeterHit.h

7.123 IMPL::RawCalorimeterHitImpl Class Reference

Implementation of the real data RawCalorimeterHit.

`#include <IMPL/RawCalorimeterHitImpl.h>`

Inheritance diagram for IMPL::RawCalorimeterHitImpl:



Public Member Functions

- **RawCalorimeterHitImpl** ()
  Default constructor, initializes values to 0.
- virtual ∼**RawCalorimeterHitImpl** ()
  Destructor.
- virtual int **id** () const
  Returns an object id for internal (debugging) use in LCIO.
- virtual int **getCellID0** () const
  Returns the detector specific (geometrical) cell id.
- virtual int **getCellID1** () const
  Returns the second detector specific (geometrical) cell id.
- virtual int **getAmplitude** () const
  Returns the amplitude of the hit in ADC counts.
- virtual int **getTimeStamp** () const
  Returns a time stamp for the hit.
- void **setCellID0** (int id0)
- void **setCellID1** (int id1)
  Sets the second cell id; Only store if the flag word (bit CHBIT_ID1) of the collection is set.
- void **setAmplitude** (int amplitude)
  Sets the amplitude.
- void **setTimeStamp** (int time)
  Set the time stamp.

Protected Attributes

- int **_cellID0**
- int **_cellID1**
- int **_amplitude**
- int **_timeStamp**

Additional Inherited Members

7.123.1 Detailed Description

Implementation of the real data RawCalorimeterHit.

Author

gaede

Version

Id:

RawCalorimeterHitImpl.h (p. **??**),v 1.3 2006-08-04 16:52:46 gaede Exp

7.123.2 Member Function Documentation

7.123.2.1 int IMPL::RawCalorimeterHitImpl::getCellID1 ( ) const [virtual]

Returns the second detector specific (geometrical) cell id.

Optional, check/set flag(LCIO::RCHBIT_ID1)==1.

Implements **EVENT::RawCalorimeterHit** (p. 189).

7.123.2.2 int IMPL::RawCalorimeterHitImpl::getTimeStamp ( ) const [virtual]

Returns a time stamp for the hit.

Optional, check/set flag(LCIO::RCHBIT_TIME)==1.

Implements **EVENT::RawCalorimeterHit** (p. 189).

7.123.2.3 void IMPL::RawCalorimeterHitImpl::setCellID1 (int *id1*)

Sets the second cell id; Only store if the flag word (bit CHBIT_ID1) of the collection is set.

Default is to store only cellid0.

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

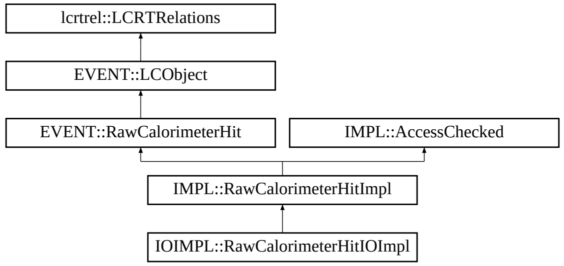
- RawCalorimeterHitImpl.h
- RawCalorimeterHitImpl.cc

7.124 IOIMPL::RawCalorimeterHitIOImpl Class Reference

Adding stuff needed for io (friend declarations, etc.)

`#include <IOIMPL/RawCalorimeterHitIOImpl.h>`

Inheritance diagram for IOIMPL::RawCalorimeterHitIOImpl:



Friends

- class **SIO::SIORawCalHitHandler**

Additional Inherited Members

7.124.1 Detailed Description

Adding stuff needed for io (friend declarations, etc.)

Author

gaede

Version

Id:

RawCalorimeterHitIOImpl.h (p. **??**),v 1.2 2005-04-15 08:37:40 gaede Exp

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

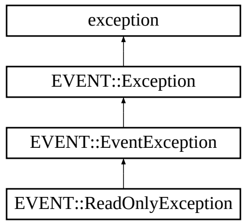
- RawCalorimeterHitIOImpl.h

7.125 EVENT::ReadOnlyException Class Reference

EventException (p. 69) used for signaling a 'read only exception'.

`#include <Exceptions.h>`

Inheritance diagram for EVENT::ReadOnlyException:



Public Member Functions

- **ReadOnlyException** (std::string text)

Additional Inherited Members

7.125.1 Detailed Description

EventException (p. 69) used for signaling a 'read only exception'.

Author

gaede

Version

Jun 5, 2003

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

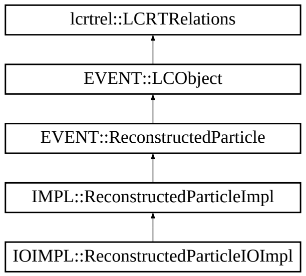
- Exceptions.h

7.126 EVENT::ReconstructedParticle Class Reference

The **LCIO** (p. 115) reconstructedParticle.

`#include <pre-generated/EVENT/ReconstructedParticle.h>`

Inheritance diagram for EVENT::ReconstructedParticle:



Public Types

- typedef **ReconstructedParticle lcobject_type**
  Useful typedef for template programming with ***LCIO*** *(p. 115).*

Public Member Functions

- virtual ∼**ReconstructedParticle** ()
  Destructor.
- virtual int **getType** () const =0

  Type of reconstructed particle.
- virtual bool **isCompound** () const =0
  Return particles primary flag.
- virtual const double ∗ **getMomentum** () const =0
  The reconstructed particle's 3-momentum - only float used in files,.
- virtual double **getEnergy** () const =0
  Energy of the reconstructed particle - only float used in files.
- virtual const **FloatVec** & **getCovMatrix** () const =0
  Covariance matrix of the reconstructed particle's 4vector (10 parameters).
- virtual double **getMass** () const =0
  Mass *(p. 169) of the reconstructed particle, set independently from four vector quantities.*
- virtual float **getCharge** () const =0
  Charge of the reconstructed particle.
- virtual const float ∗ **getReferencePoint** () const =0
  Reference point of the reconstructedParticle parameters.
- virtual const **ParticleIDVec** & **getParticleIDs** () const =0
  The particle Id's sorted by their likelihood.
- virtual **ParticleID** ∗ **getParticleIDUsed** () const =0
  The particle Id used for the kinematics of this particle.
- virtual float **getGoodnessOfPID** () const =0
  The overall goodness of the PID on a scale of [0;1].
- virtual const **ReconstructedParticleVec** & **getParticles** () const =0
  The reconstructed particles that have been combined to this particle.
- virtual const **ClusterVec** & **getClusters** () const =0
  The weights of the reconstructed particles combined to this particle.
- virtual const **TrackVec** & **getTracks** () const =0
  The weights of cluster contributions to this particle.
- virtual **Vertex** ∗ **getStartVertex** () const =0
  The start vertex associated to this particle.
- virtual **Vertex** ∗ **getEndVertex** () const =0
  The vertex where the particle decays This method actually returns the start vertex from the first daughter particle found.
- virtual void **addParticleID** (**ParticleID** ∗pid)=0
  Set the type of reconstructed particle, one of:
  ReconstructedParticle::SINGLE,
  ReconstructedParticle::V0,
  ReconstructedParticle::COMPOUND,
  ReconstructedParticle::JET
- virtual void **addParticle** (**ReconstructedParticle** ∗particle)=0
  Add a particle that has been used to create this particle.
- virtual void **addCluster** (**Cluster** ∗cluster)=0
  Add a cluster that has been used to create this particle.
- virtual void **addTrack** (**Track** ∗track)=0
  Add a track that has been used to create this particle.

Additional Inherited Members

7.126.1 Detailed Description

The **LCIO** (p. 115) reconstructedParticle.

Author

gaede

Version

Id:

ReconstructedParticle.aid,v 1.18 2006-09-21 06:10:35 gaede Exp

7.126.2 Member Function Documentation

7.126.2.1 virtual void EVENT::ReconstructedParticle::addParticleID (ParticleID ∗ *pid*) [pure virtual]

Set the type of reconstructed particle, one of:

ReconstructedParticle::SINGLE,

ReconstructedParticle::V0,

ReconstructedParticle::COMPOUND,

ReconstructedParticle::JET

Set primary flag.Set particle momentum (px,py,pz).Set energy.Set the values of the covariance matrix of the particles four vector (px,py,pz,E).Set the values of the covariance matrix of the particles four vector (px,py,pz,E).Set mass - independently from four vector. Set the charge. Set the reference point, i.e. the point where the four vector is defined.Add a **ParticleID** (p. 181) object.

See Also

ParticleID (p. 181)

Implemented in **IMPL::ReconstructedParticleImpl** (p. 199).

7.126.2.2 virtual const ClusterVec& EVENT::ReconstructedParticle::getClusters ( ) const [pure virtual]

The weights of the reconstructed particles combined to this particle.

The clusters that have been used for this particle.

Implemented in **IMPL::ReconstructedParticleImpl** (p. 198).

Referenced by SIO::SIOReconstructedParticleHandler::write().

7.126.2.3 virtual const FloatVec& EVENT::ReconstructedParticle::getCovMatrix ( ) const [pure virtual]

Covariance matrix of the reconstructed particle's 4vector (10 parameters).

Stored as lower triangle matrix of the four momentum (px,py,pz,E), i.e. cov(px,px), cov(py,px), cov(py,py) , ...

Implemented in **IMPL::ReconstructedParticleImpl** (p. 197).

Referenced by SIO::SIOReconstructedParticleHandler::write().

7.126.2.4 virtual double EVENT::ReconstructedParticle::getMass ( ) const [pure virtual]

Mass (p. 169) of the reconstructed particle, set independently from four vector quantities.

- only float used in files.

Implemented in **IMPL::ReconstructedParticleImpl** (p. 197).

Referenced by SIO::SIOReconstructedParticleHandler::write().

7.126.2.5 virtual const ParticleIDVec& EVENT::ReconstructedParticle::getParticleIDs ( ) const [pure virtual]

The particle Id's sorted by their likelihood.

See Also

ParticleID (p. 181)

Implemented in **IMPL::ReconstructedParticleImpl** (p. 199).

Referenced by UTIL::LCTOOLS::printReconstructedParticles(), and SIO::SIOReconstructedParticleHandler::write().

7.126.2.6 virtual ParticleID∗ EVENT::ReconstructedParticle::getParticleIDUsed ( ) const [pure virtual]

The particle Id used for the kinematics of this particle.

See Also

ParticleID (p. 181)

Implemented in **IMPL::ReconstructedParticleImpl** (p. 200).

Referenced by SIO::SIOReconstructedParticleHandler::write().

7.126.2.7 virtual const TrackVec& EVENT::ReconstructedParticle::getTracks ( ) const [pure virtual]

The weights of cluster contributions to this particle.

The tracks that have been used for this particle.

Implemented in **IMPL::ReconstructedParticleImpl** (p. 198).

Referenced by SIO::SIOReconstructedParticleHandler::write().

7.126.2.8 virtual int EVENT::ReconstructedParticle::getType ( ) const [pure virtual]

Type of reconstructed particle.

Check/set collection parameters ReconstructedParticleTypeNames and ReconstructedParticleTypeValues.

Implemented in **IMPL::ReconstructedParticleImpl** (p. 200).

Referenced by SIO::SIOReconstructedParticleHandler::write().

7.126.2.9 virtual bool EVENT::ReconstructedParticle::isCompound ( ) const [pure virtual]

Return particles primary flag.

All particles in the **ReconstructedParticle** (p. 193) collection should by definition return true. Compound partciles will return false.Convenient method - same as (**getParticles()** (p. 194).size() > 0).

Implemented in **IMPL::ReconstructedParticleImpl** (p. 197).

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

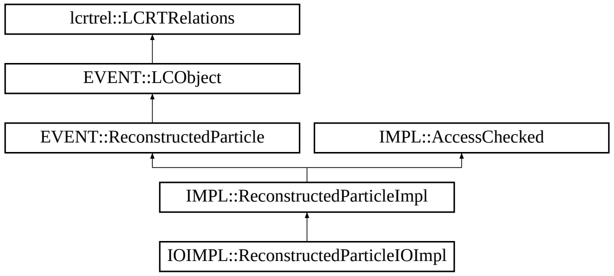
- ReconstructedParticle.h

7.127 IMPL::ReconstructedParticleImpl Class Reference

Implementation of ReconstructedParticle.

`#include <IMPL/ReconstructedParticleImpl.h>`

Inheritance diagram for IMPL::ReconstructedParticleImpl:



Public Member Functions

- **ReconstructedParticleImpl** ()
  Default constructor, initializes values to 0.
- virtual ∼**ReconstructedParticleImpl** ()
  Destructor.
- virtual int **id** () const
  Returns an object id for internal (debugging) use in LCIO.
- virtual int **getType** () const
  Type of reconstructed particle.
- virtual bool **isCompound** () const
  *Convenient method - same as (**getParticles()** (p. 198).size() > 0).*
- virtual const double ∗ **getMomentum** () const
  The magnitude of the reconstructed particle's momentum,.
- virtual double **getEnergy** () const
  Energy of the reconstructed particle.
- virtual const **EVENT::FloatVec** & **getCovMatrix** () const
  Covariance matrix of the reconstructed particle's 4vector (10 parameters).
- virtual double **getMass** () const
  ***Mass** (p. 169) of the reconstructed particle, set independently from four vector quantities.*
- virtual float **getCharge** () const
  Charge of the reconstructed particle.
- virtual const float ∗ **getReferencePoint** () const
  Reference point of the reconstructedParticle parameters.

- virtual const
  EVENT::ParticleIDVec & **getParticleIDs** () const
  The particle Id's sorted by their likelihood.
- virtual **EVENT::ParticleID** ∗ **getParticleIDUsed** () const
  The particle Id used for the kinematics of this particle.
- virtual float **getGoodnessOfPID** () const
  The overall goodness of the PID on a scale of [0;1].
- virtual const
  EVENT::ReconstructedParticleVec & **getParticles** () const
  The reconstructed particles that have been combined to this particle.
- virtual const **EVENT::ClusterVec** & **getClusters** () const
  The clusters that have been used for this particle.
- virtual const **EVENT::TrackVec** & **getTracks** () const
  The tracks that have been used for this particle.
- virtual **EVENT::Vertex** ∗ **getStartVertex** () const
  The start vertex associated to this particle.
- virtual **EVENT::Vertex** ∗ **getEndVertex** () const
  The vertex where the particle decays This method actually returns the start vertex from the first daughter particle found.
- void **setType** (int type)
- void **setMomentum** (const float ∗momentum)
- void **setMomentum** (const double ∗momentum)
- void **setEnergy** (float energy)
- void **setCovMatrix** (const float ∗cov)
- void **setCovMatrix** (const **EVENT::FloatVec** &)
- void **setMass** (float mass)
- void **setCharge** (float charge)
- void **setReferencePoint** (const float ∗reference)
- void **addParticleID** (**EVENT::ParticleID** ∗pid)
  Set the type of reconstructed particle, one of:
  ReconstructedParticle::SINGLE,
  ReconstructedParticle::V0,
  ReconstructedParticle::COMPOUND,
  ReconstructedParticle::JET
- void **setParticleIDUsed** (**EVENT::ParticleID** ∗pid)
- void **setGoodnessOfPID** (float goodness)
- void **addParticle** (**EVENT::ReconstructedParticle** ∗particle)
  Add a particle that has been used to create this particle.
- void **addCluster** (**EVENT::Cluster** ∗cluster)
  Add a cluster that has been used to create this particle.
- void **addTrack** (**EVENT::Track** ∗track)
  Add a track that has been used to create this particle.
- void **setStartVertex** (**EVENT::Vertex** ∗sv)

Protected Attributes

- int **_type**
- double **_momentum** [3]
- double **_energy**
- **EVENT::FloatVec _cov**
- double **_mass**
- float **_charge**
- float **_reference** [3]

- **EVENT::ParticleID ∗ _pidUsed**
- float **_goodnessOfPID**
- **EVENT::ParticleIDVec _pid**
- **EVENT::ReconstructedParticleVec _particles**
- **EVENT::ClusterVec _clusters**
- **EVENT::TrackVec _tracks**
- **EVENT::Vertex ∗ _sv**

Additional Inherited Members

7.127.1 Detailed Description

Implementation of ReconstructedParticle.

See Also

ReconstructedParticle

Author

gaede

Version

Mar 30, 2003

7.127.2 Member Function Documentation

7.127.2.1 void IMPL::ReconstructedParticleImpl::addParticleID (EVENT::ParticleID ∗ *pid*) [virtual]

Set the type of reconstructed particle, one of:

ReconstructedParticle::SINGLE,

ReconstructedParticle::V0,

ReconstructedParticle::COMPOUND,

ReconstructedParticle::JET

Set primary flag.Set particle momentum (px,py,pz).Set energy.Set the values of the covariance matrix of the particles four vector (px,py,pz,E).Set the values of the covariance matrix of the particles four vector (px,py,pz,E).Set mass - independently from four vector. Set the charge. Set the reference point, i.e. the point where the four vector is defined.Add a ParticleID object.

See Also

ParticleID

Implements **EVENT::ReconstructedParticle** (p. 195).

Referenced by SIO::SIOReconstructedParticleHandler::read().

7.127.2.2 const EVENT::ParticleIDVec & IMPL::ReconstructedParticleImpl::getParticleIDs ( ) const [virtual]

The particle Id's sorted by their likelihood.

See Also

ParticleID

Implements **EVENT::ReconstructedParticle** (p. 196).

Referenced by UTIL::PIDHandler::setParticleIDUsed().

7.127.2.3 EVENT::ParticleID ∗ IMPL::ReconstructedParticleImpl::getParticleIDUsed ( ) const [virtual]

The particle Id used for the kinematics of this particle.

See Also

ParticleID

Implements **EVENT::ReconstructedParticle** (p. 196).

7.127.2.4 int IMPL::ReconstructedParticleImpl::getType ( ) const [virtual]

Type of reconstructed particle.

Check/set collection parameterrs ReconstructedParticleTypeNames and ReconstructedParticleTypeValues.

Implements **EVENT::ReconstructedParticle** (p. 196).

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

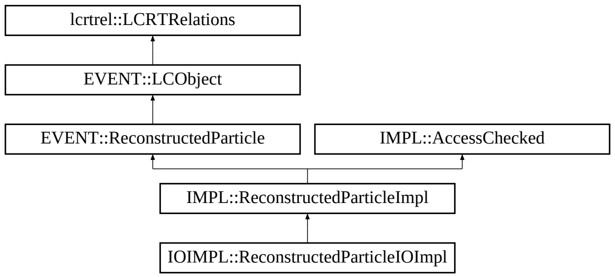
- ReconstructedParticleImpl.h
- ReconstructedParticleImpl.cc

7.128 IOIMPL::ReconstructedParticleIOImpl Class Reference

Adding stuff needed for io (friend declarations, etc.)

`#include <IOIMPL/ReconstructedParticleIOImpl.h>`

Inheritance diagram for IOIMPL::ReconstructedParticleIOImpl:



Friends

- class **SIO::SIOReconstructedParticleHandler**

Additional Inherited Members

7.128.1 Detailed Description

Adding stuff needed for io (friend declarations, etc.)

Author

gaede

Version

Mar 31, 2004

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

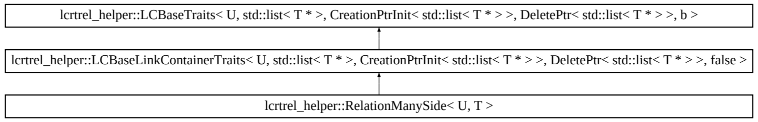
- ReconstructedParticleIOImpl.h

7.129 lcrtrel_helper::RelationManySide< U, T > Struct Template Reference

Helper class for relations.

```
#include <LCRTRelations.h>
```

Inheritance diagram for lcrtrel_helper::RelationManySide< U, T >:



Additional Inherited Members

7.129.1 Detailed Description

template< class U, class T > struct lcrtrel_helper::RelationManySide< U, T >

Helper class for relations.

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

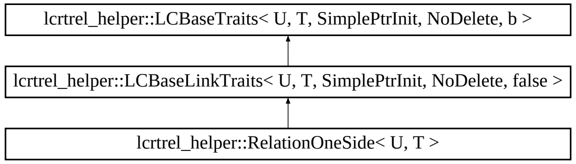
- LCRTRelations.h

7.130 lcrtrel_helper::RelationOneSide< U, T > Struct Template Reference

Helper class for relations.

```
#include <LCRTRelations.h>
```

Inheritance diagram for lcrtrel_helper::RelationOneSide< U, T >:



Additional Inherited Members

7.130.1 Detailed Description

template< class U, class T > struct lcrtrel_helper::RelationOneSide< U, T >

Helper class for relations.

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

- LCRTRelations.h

7.131 SIO::RunEvent Struct Reference

Helper struct that stores run and event positions in the file.

```
#include <SIO/LCIORandomAccess.h>
```

Public Member Functions

- **RunEvent** (int run, int evt)
- **RunEvent** (long64 runEvt)
- **operator long64** () const

Public Attributes

- int **RunNum**
- int **EvtNum**

7.131.1 Detailed Description

Helper struct that stores run and event positions in the file.

The operator<() defines a lexicographical ordering in RunNum and EvtNum where all run records (EvtNum=-1) are orderd first.

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

- LCIORandomAccess.h

7.132 SIO::RunEventMap Class Reference

Map that holds positions of Run and Event records.

```
#include <SIO/RunEventMap.h>
```

Public Types

- typedef std::map< **RunEvent**, long64 > **MapType**
- typedef MapType::iterator **Map_IT**
- typedef MapType::const_iterator **Map_cIT**

Public Member Functions

- void **add** (const **RunEvent** &re, long64 pos)

  *Add a new entry to the event map - if the **RunEvent** (p. 202) already exists the new position will be stored.*
- long64 **getPosition** (const **RunEvent** &re)

  Return the position of the specified Event record or Run record respectively (if EventNum == -1).
- long64 **getPosition** (long64 re)

  *Same as **getPosition(const RunEvent& re)** (p. 204).*
- **RunEvent minRunEvent** ()
- **RunEvent maxRunEvent** ()
- int **getNumberOfEventRecords** () const
- int **getNumberOfRunRecords** () const
- size_t **size** ()
- Map_IT **begin** ()
- Map_IT **end** ()
- Map_cIT **begin** () const
- Map_cIT **end** () const
- bool **empty** () const
- template<class InputIterator >
  void **insert** (InputIterator b, InputIterator e)
- void **clear** ()

Static Public Attributes

- static const int **NPos** = -1

Protected Attributes

- MapType **_map**
- int **_nRun**
- int **_nEvt**

Friends

- std::ostream & **operator**<< (std::ostream &os, const **RunEventMap** &rm)

7.132.1 Detailed Description

Map that holds positions of Run and Event records.

Author

gaede

Version

Id:

RunEventMap.h (p. **??**),v 1.3 2011-03-03 16:00:12 gaede Exp

7.132.2 Member Function Documentation

7.132.2.1 long64 SIO::RunEventMap::getPosition (const RunEvent & *re*) [inline]

Return the position of the specified Event record or Run record respectively (if EventNum == -1).

Returns RunEventMap::NPos if no record found.

Referenced by SIO::LCIORandomAccessMgr::getPosition().

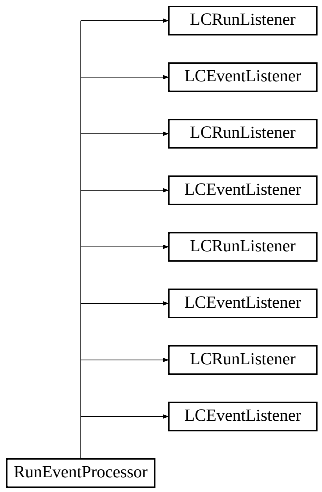
The documentation for this class was generated from the following files:

- RunEventMap.h
- RunEventMap.cc

7.133 RunEventProcessor Class Reference

Little tool that copies LCIO files on an event by event and run by run basis, thus fixing files that have incomplete records at the end due to a program crash.

Inheritance diagram for RunEventProcessor:



Public Member Functions

- **RunEventProcessor** (const char ∗outFileName)
- void **modifyEvent** (**LCEvent** ∗evt)
- void **processEvent** (**LCEvent** ∗evt)

- void **modifyRunHeader** (**LCRunHeader** ∗run)
- void **processRunHeader** (**LCRunHeader** ∗run)
- **RunEventProcessor** (const char ∗outFileName, unsigned int nFiles)
- void **modifyEvent** (**LCEvent** ∗evt)
- void **processEvent** (**LCEvent** ∗evt)
- void **modifyRunHeader** (**LCRunHeader** ∗run)
- void **processRunHeader** (**LCRunHeader** ∗run)
- **RunEventProcessor** (const char ∗outFileName, int splitSize)
- void **modifyEvent** (**LCEvent** ∗evt)
- void **processEvent** (**LCEvent** ∗evt)
- void **modifyRunHeader** (**LCRunHeader** ∗run)
- void **processRunHeader** (**LCRunHeader** ∗run)
- void **processEvent** (**LCEvent** ∗evt)
- void **modifyEvent** (**LCEvent** ∗evt)
- void **processRunHeader** (**LCRunHeader** ∗run)
- void **modifyRunHeader** (**LCRunHeader** ∗run)

Protected Attributes

- **LCWriter** ∗ **lcWrt**
- int **nEvent**
- unsigned int **_nFiles**

7.133.1 Detailed Description

Little tool that copies LCIO files on an event by event and run by run basis, thus fixing files that have incomplete records at the end due to a program crash.

Example of reading events from the file, add sth.

Little tool that splits large LCIO files.

lcio tool for merging files on an event by event and run by run basis

to the event and write it to a new file. This uses the listener mechanism to read the input file (run headers and events) record by record.

The **RunEventProcessor** (p. 204) class is defined for processing run and event records. This is our analysis module. For simplicity it is defined in the same file - in a real world application it should of course be defined in separate header and source files.

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

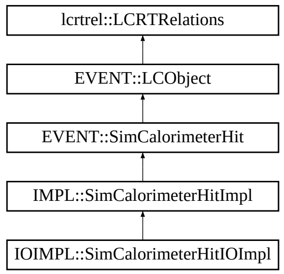
- copyfix.cc
- lcio_merge_files.cc
- lcio_split_file.cc
- recjob.cc

7.134 EVENT::SimCalorimeterHit Class Reference

The generic simulated calorimeter hit.

`#include <pre-generated/EVENT/SimCalorimeterHit.h>`

Inheritance diagram for EVENT::SimCalorimeterHit:



Public Types

- typedef **SimCalorimeterHit lcobject_type**

  Useful typedef for template programming with ***LCIO*** *(p. 115).*

Public Member Functions

- virtual ∼**SimCalorimeterHit** ()

  Destructor.
- virtual int **getCellID0** () const =0

  Returns the detector specific (geometrical) cell id.
- virtual int **getCellID1** () const =0

  Returns the second detector specific (geometrical) cell id.
- virtual float **getEnergy** () const =0

  Returns the energy of the hit in [GeV].
- virtual const float ∗ **getPosition** () const =0

  Returns the position of the hit in world coordinates.
- virtual const float ∗ **getStepPosition** (int i) const =0

  Returns the position where the energy deposited (step) occurred optional, only if bit LCIO::CHBIT_STEP is set.
- virtual int **getNMCParticles** () const =0

  Returns the number of MC contributions to the hit.
- virtual int **getNMCContributions** () const =0

  Returns the number of MC contributions to the hit.
- virtual float **getEnergyCont** (int i) const =0

  Returns the energy in [GeV] of the i-th contribution to the hit.
- virtual float **getTimeCont** (int i) const =0

  Returns the time of the i-th in [ns] contribution to the hit.
- virtual int **getPDGCont** (int i) const =0

  Returns the PDG code of the shower particle that caused this contribution.
- virtual **MCParticle** ∗ **getParticleCont** (int i) const =0

  Returns the ***MCParticle*** *(p. 169) that caused the shower responsible for this contribution to the hit.*

Additional Inherited Members

7.134.1 Detailed Description

The generic simulated calorimeter hit.

Author

gaede

Version

Id:

SimCalorimeterHit.aid,v 1.8 2006-03-24 13:25:52 gaede Exp

7.134.2 Member Function Documentation

7.134.2.1 virtual int EVENT::SimCalorimeterHit::getCellID1 ( ) const [pure virtual]

Returns the second detector specific (geometrical) cell id.

0 if information is not stored - check the flag word (bit CHBIT_ID1) of the collection. Default is to store only cellid0.

Implemented in **IMPL::SimCalorimeterHitImpl** (p. 211).

Referenced by SIO::SIOSimCalHitHandler::write().

7.134.2.2 virtual float EVENT::SimCalorimeterHit::getEnergyCont (int *i*) const [pure virtual]

Returns the energy in [GeV] of the i-th contribution to the hit.

See Also

getNMCContributions() (p. 207)

Implemented in **IMPL::SimCalorimeterHitImpl** (p. 211).

Referenced by SIO::SIOSimCalHitHandler::write().

7.134.2.3 virtual int EVENT::SimCalorimeterHit::getNMCContributions ( ) const [pure virtual]

Returns the number of MC contributions to the hit.

0 if information is not stored. There are two levels of detail: if collection flag bit LCIO.CHBIT_STEP==1 then all simulator steps' contributions to the hit are stored, otherwise there is only one contribution for every particle entering the calorimeter.

Implemented in **IMPL::SimCalorimeterHitImpl** (p. 211).

Referenced by SIO::SIOSimCalHitHandler::write().

7.134.2.4 virtual int EVENT::SimCalorimeterHit::getNMCParticles ( ) const [pure virtual]

Returns the number of MC contributions to the hit.

0 if information is not stored. Renamed to getNMCContributions.

See Also

Deprecated **getNMCContributions()** (p. 207)

Implemented in **IMPL::SimCalorimeterHitImpl** (p. 211).

7.134.2.5 virtual MCParticle* EVENT::SimCalorimeterHit::getParticleCont (int *i*) const [pure virtual]

Returns the **MCParticle** (p. 169) that caused the shower responsible for this contribution to the hit.

This is the particle that flew into the calorimeter and not the shower particle that made the energy deposition.

See Also

getNMCContributions() (p. 207)

Implemented in **IMPL::SimCalorimeterHitImpl** (p. 211).

Referenced by SIO::SIOSimCalHitHandler::write().

7.134.2.6 virtual int EVENT::SimCalorimeterHit::getPDGCont (int *i*) const [pure virtual]

Returns the PDG code of the shower particle that caused this contribution.

Check the flag word bit LCIO.CHBIT_STEP of the collection whether this information is available.

See Also

getNMCContributions() (p. 207)

Implemented in **IMPL::SimCalorimeterHitImpl** (p. 212).

Referenced by SIO::SIOSimCalHitHandler::write().

7.134.2.7 virtual const float* EVENT::SimCalorimeterHit::getPosition ( ) const [pure virtual]

Returns the position of the hit in world coordinates.

NULL if information is not stored - check the flag word (bit CHBIT_LONG) of the collection.

Implemented in **IMPL::SimCalorimeterHitImpl** (p. 212).

Referenced by SIO::SIOSimCalHitHandler::write().

7.134.2.8 virtual float EVENT::SimCalorimeterHit::getTimeCont (int *i*) const [pure virtual]

Returns the time of the i-th in [ns] contribution to the hit.

See Also

getNMCContributions() (p. 207)

Implemented in **IMPL::SimCalorimeterHitImpl** (p. 212).

Referenced by SIO::SIOSimCalHitHandler::write().

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

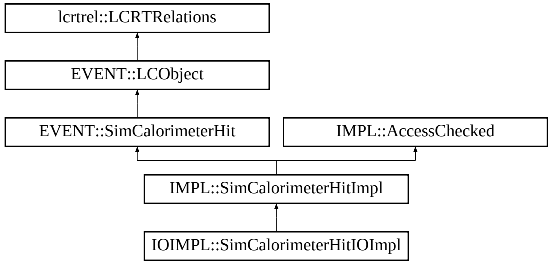
- SimCalorimeterHit.h

7.135 IMPL::SimCalorimeterHitImpl Class Reference

Implementation of the generic SimCalorimeterHit.

`#include <IMPL/SimCalorimeterHitImpl.h>`

Inheritance diagram for IMPL::SimCalorimeterHitImpl:



Public Member Functions

- **SimCalorimeterHitImpl** ()

  Default constructor, initializes values to 0.
- **SimCalorimeterHitImpl** (const **EVENT::SimCalorimeterHit** &hit)

  Copy constructor - shallow copy.
- virtual ∼**SimCalorimeterHitImpl** ()

  Destructor.
- virtual int **id** () const

  Returns an object id for internal (debugging) use in LCIO.
- virtual int **getCellID0** () const

  Create a deep copy of itself.
- virtual int **getCellID1** () const

  Returns the second detector specific (geometrical) cell id.
- virtual float **getEnergy** () const

  Returns the energy of the hit.
- virtual const float ∗ **getPosition** () const

  Returns the position of the hit in world coordinates.
- virtual const float ∗ **getStepPosition** (int i) const

  Returns the position where the energy deposited (step) occurred optional, only if bit LCIO::CHBIT_STEP is set.
- virtual int **getNMCParticles** () const

  Returns the number of MC contributions to the hit.
- virtual int **getNMCContributions** () const

  Returns the number of MC contributions to the hit.
- virtual float **getEnergyCont** (int i) const

  Returns the energy in [GeV] of the i-th contribution to the hit.
- virtual float **getTimeCont** (int i) const

  Returns the time of the i-th in [ns] contribution to the hit.
- virtual int **getPDGCont** (int i) const

  Returns the PDG code of the shower particle that caused this contribution.
- virtual **EVENT::MCParticle** ∗ **getParticleCont** (int i) const

  Returns the MCParticle that caused the shower responsible for this contribution to the hit.
- void **setCellID0** (int id0)

  Sets the first cell id;.
- void **setCellID1** (int id1)

  Sets the second cell id; Only store if the flag word (bit CHBIT_ID1) of the collection is set.
- void **setEnergy** (float en)

  Sets the energy.

- void **setPosition** (const float pos[3])

  Sets the position.
- void **addMCParticleContribution** (**EVENT::MCParticle** ∗p, float en, float t)

  Adds an MCParticle contribution to the hit - contributions for the same MCParticle are combined into one, where the energy is accumulated and the time is the energy weighted mean time.
- void **addMCParticleContribution** (**EVENT::MCParticle** ∗p, float en, float t, int pdg, float ∗stepPos=0)

  Adds a detailed MCParticle contribution to the hit.

Protected Attributes

- int **_cellID0**
- int **_cellID1**
- float **_energy**
- float **_position** [3]
- MCParticleContVec **_vec**

Additional Inherited Members

7.135.1 Detailed Description

Implementation of the generic SimCalorimeterHit.

Author

gaede

Version

Mar 5, 2003

7.135.2 Member Function Documentation

7.135.2.1 void IMPL::SimCalorimeterHitImpl::addMCParticleContribution (EVENT::MCParticle ∗ *p*, float *en*, float *t*)

Adds an MCParticle contribution to the hit - contributions for the same MCParticle are combined into one, where the energy is accumulated and the time is the energy weighted mean time.

Use this method for the standard LCIO mode, where one energy contribution per primary particle entering the calorimeter is stored.

7.135.2.2 void IMPL::SimCalorimeterHitImpl::addMCParticleContribution (EVENT::MCParticle ∗ *p*, float *en*, float *t*, int *pdg*, float ∗ *stepPos* = 0)

Adds a detailed MCParticle contribution to the hit.

This method should be used for the detailed mode, where one MCParticleContribution is stored for every simulator step.

If stepPos==0, (0,0,0) will be stored. NB: The flag word bit LCIO::CHBIT_STEP (or LCIO::CHBIT_PDG) has to be set, in order for the PDG and step position to be stored.

7.135.2.3 int IMPL::SimCalorimeterHitImpl::getCellID0 ( ) const [virtual]

Create a deep copy of itself.

Overwrites return type in LCObject::clone(). Not yet - as AID genereates virtual inheritance and g++ doesn't support virtual constructors with virtual inheritance TODO - change org.freehep.aid.CPPHeaderGenerator !Returns the detector specific (geometrical) cell id.

Implements **EVENT::SimCalorimeterHit** (p. 206).

7.135.2.4 int IMPL::SimCalorimeterHitImpl::getCellID1 ( ) const [virtual]

Returns the second detector specific (geometrical) cell id.

0 if information is not stored - check the flag word (bit CHBIT_ID1) of the collection. Default is to store only cellid0.

Implements **EVENT::SimCalorimeterHit** (p. 207).

7.135.2.5 float IMPL::SimCalorimeterHitImpl::getEnergyCont (int *i*) const [virtual]

Returns the energy in [GeV] of the i-th contribution to the hit.

See Also

getNMCContributions() (p. 211)

Implements **EVENT::SimCalorimeterHit** (p. 207).

7.135.2.6 int IMPL::SimCalorimeterHitImpl::getNMCContributions ( ) const [virtual]

Returns the number of MC contributions to the hit.

0 if information is not stored. There are two levels of detail: if collection flag bit LCIO.CHBIT_STEP==1 then all simulator steps' contributions to the hit are stored, otherwise there is only one contribution for every particle entering the calorimeter.

Implements **EVENT::SimCalorimeterHit** (p. 207).

Referenced by getNMCParticles().

7.135.2.7 int IMPL::SimCalorimeterHitImpl::getNMCParticles ( ) const [virtual]

Returns the number of MC contributions to the hit.

0 if information is not stored. Renamed to getNMCContributions.

See Also

Deprecated **getNMCContributions()** (p. 211)

Implements **EVENT::SimCalorimeterHit** (p. 207).

References getNMCContributions(), and UTIL::LCWarning::printWarning().

7.135.2.8 EVENT::MCParticle ∗ IMPL::SimCalorimeterHitImpl::getParticleCont (int *i*) const [virtual]

Returns the MCParticle that caused the shower responsible for this contribution to the hit.

This is the particle that flew into the calorimeter and not the shower particle that made the energy deposition.

See Also

getNMCContributions() (p. 211)

Implements **EVENT::SimCalorimeterHit** (p. 208).

7.135.2.9 int IMPL::SimCalorimeterHitImpl::getPDGCont (int *i*) const [virtual]

Returns the PDG code of the shower particle that caused this contribution.

Check the flag word bit LCIO.CHBIT_STEP of the collection whether this information is available.

See Also

getNMCContributions() (p. 211)

Implements **EVENT::SimCalorimeterHit** (p. 208).

7.135.2.10 const float ∗ IMPL::SimCalorimeterHitImpl::getPosition ( ) const [virtual]

Returns the position of the hit in world coordinates.

NULL if information is not stored. Ask collection for flag, only available if bit LCIO.CHBIT_LONG is set.

Implements **EVENT::SimCalorimeterHit** (p. 208).

7.135.2.11 float IMPL::SimCalorimeterHitImpl::getTimeCont (int *i*) const [virtual]

Returns the time of the i-th in [ns] contribution to the hit.

See Also

getNMCContributions() (p. 211)

Implements **EVENT::SimCalorimeterHit** (p. 208).

7.135.2.12 void IMPL::SimCalorimeterHitImpl::setCellID1 (int *id1*)

Sets the second cell id; Only store if the flag word (bit CHBIT_ID1) of the collection is set.

Default is to store only cellid0.

7.135.2.13 void IMPL::SimCalorimeterHitImpl::setPosition (const float *pos[3]*)

Sets the position.

Not stored if LCIO.CHBIT_LONG isn't set.

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

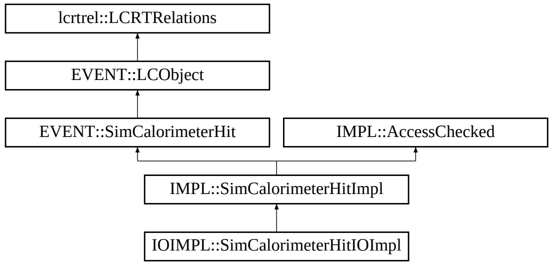
- SimCalorimeterHitImpl.h
- SimCalorimeterHitImpl.cc

7.136 IOIMPL::SimCalorimeterHitIOImpl Class Reference

Adding stuff needed for io (friend declarations, etc.)

`#include <IOIMPL/SimCalorimeterHitIOImpl.h>`

Inheritance diagram for IOIMPL::SimCalorimeterHitIOImpl:



Friends

- class **SIO::SIOSimCalHitHandler**

Additional Inherited Members

7.136.1 Detailed Description

Adding stuff needed for io (friend declarations, etc.)

Author

gaede

Version

Mar 7, 2003

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

- SimCalorimeterHitIOImpl.h

7.137 lcrtrel_helper::SimplePtrInit Struct Reference

Simple init function for simple pointers.

`#include <LCRTRelations.h>`

Static Public Member Functions

- static void ∗ **init** ()

7.137.1 Detailed Description

Simple init function for simple pointers.

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

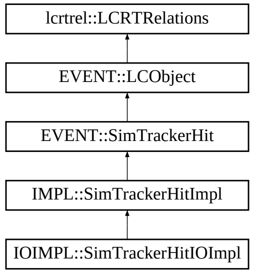
- LCRTRelations.h

7.138 EVENT::SimTrackerHit Class Reference

A generic simulated tracker hit.

`#include <pre-generated/EVENT/SimTrackerHit.h>`

Inheritance diagram for EVENT::SimTrackerHit:



Public Types

- typedef **SimTrackerHit lcobject_type**

  Useful typedef for template programming with ***LCIO*** *(p. 115).*

Public Member Functions

- virtual ∼**SimTrackerHit** ()

  Destructor.
- virtual int **getCellID0** () const =0

  Same name as in ***CalorimeterHit*** *(p. 49), even though there are no 'cells' in this case.*
- virtual int **getCellID1** () const =0

  Same name as in ***CalorimeterHit*** *(p. 49), even though there are no 'cells' in this case Optional, check/set flag(LCIO-::THBIT_ID1)==1.*
- virtual int **getCellID** () const =0

  DEPRECATED: use ()
- virtual const double ∗ **getPosition** () const =0

  Returns the hit position in [mm].
- virtual float **getdEdx** () const =0

  Returns the dE/dx of the hit in [GeV].
- virtual float **getEDep** () const =0

  The energy deposited on the hit [GeV].
- virtual float **getTime** () const =0

  Returns the time of the hit in [ns].
- virtual **MCParticle** ∗ **getMCParticle** () const =0

  Returns the MC particle that caused the hit.
- virtual const float ∗ **getMomentum** () const =0

  Returns the 3-momentum of the particle at the hits position in [GeV] - optional, only if bit LCIO::THBIT_MOMENTUM is set.
- virtual float **getPathLength** () const =0

  The path length of the particle in the sensitive material that resulted in this hit.

Additional Inherited Members

7.138.1 Detailed Description

A generic simulated tracker hit.

Author

gaede

Version

Id:

SimTrackerHit.aid,v 1.11 2010-06-17 12:28:59 engels Exp

7.138.2 Member Function Documentation

7.138.2.1 virtual float EVENT::SimTrackerHit::getdEdx ( ) const [pure virtual]

Returns the dE/dx of the hit in [GeV].

See Also

Deprecated **getEDep()** (p. 214)

Implemented in **IMPL::SimTrackerHitImpl** (p. 218).

7.138.2.2 virtual MCParticle∗ EVENT::SimTrackerHit::getMCParticle ( ) const [pure virtual]

Returns the MC particle that caused the hit.

See Also

MCParticle (p. 169)

Implemented in **IMPL::SimTrackerHitImpl** (p. 218).

Referenced by SIO::SIOSimTrackHitHandler::write().

7.138.2.3 virtual float EVENT::SimTrackerHit::getPathLength ( ) const [pure virtual]

The path length of the particle in the sensitive material that resulted in this hit.

This is only stored together with momentum, i.e. if LCIO::THBIT_MOMENTUM is set.

Implemented in **IMPL::SimTrackerHitImpl** (p. 218).

Referenced by SIO::SIOSimTrackHitHandler::write().

7.138.2.4 virtual float EVENT::SimTrackerHit::getTime ( ) const [pure virtual]

Returns the time of the hit in [ns].

TO DO needs definition.

Implemented in **IMPL::SimTrackerHitImpl** (p. 218).

Referenced by SIO::SIOSimTrackHitHandler::write().

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

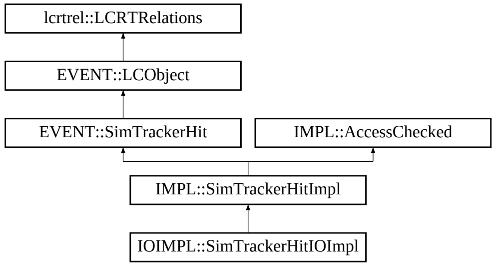
- SimTrackerHit.h

7.139 IMPL::SimTrackerHitImpl Class Reference

Implementation of SimTrackerHit.

`#include <IMPL/SimTrackerHitImpl.h>`

Inheritance diagram for IMPL::SimTrackerHitImpl:



Public Member Functions

- **SimTrackerHitImpl** ()
  Default constructor, initializes values to 0.
- virtual ∼**SimTrackerHitImpl** ()
  Destructor.
- virtual int **id** () const
  Returns an object id for internal (debugging) use in LCIO.
- virtual int **getCellID** () const
  DEPRECATED: use ()
- virtual int **getCellID0** () const
  Same name as in CalorimeterHit, even though there are no 'cells' in this case.
- virtual int **getCellID1** () const
  Same name as in CalorimeterHit, even though there are no 'cells' in this case 0 if information is not stored - check the flag word (bit THBIT_ID1) of the collection.
- virtual const double ∗ **getPosition** () const
  Returns the hit position.
- virtual float **getdEdx** () const
  Returns the dE/dx of the hit.
- virtual float **getEDep** () const
  Returns the deposited energy of the hit [GeV].
- virtual float **getTime** () const
  Returns the time of the hit.
- virtual **EVENT::MCParticle** ∗ **getMCParticle** () const
  Returns the MC particle that caused the hit.
- virtual const float ∗ **getMomentum** () const

Returns the 3-momentum of the particle at the hits position in [GeV] - optional, only if bit LCIO::THBIT_MOMENTUM is set.

- virtual float **getPathLength** () const
  The path length of the particle in the sensitive material that resulted in this hit.
- void **setCellID** (int **id**)
  DEPRECATED: use .
- void **setCellID0** (int id0)
  Sets the first cell id;.
- void **setCellID1** (int id1)
  Sets the second cell id; Only store if the flag word (bit THBIT_ID1) of the collection is set.
- void **setPosition** (const double pos[3])
  Sets the position.
- void **setdEdx** (float dEdX)
  Sets dE/dx.
- void **setEDep** (float e)
  Sets EDep.
- void **setTime** (float t)
  Sets the time.
- void **setMCParticle** (**EVENT::MCParticle** ∗particle)
  Sets the MCParticle that caused the hit.
- void **setMomentum** (const float p[3])
  Sets the momentum of the particle at the hit's position.
- void **setMomentum** (float px, float py, float pz)
  Sets the momentum of the particle at the hit's position.
- void **setPathLength** (float pathLength)

Protected Attributes

- int **_cellID0**
- int **_cellID1**
- double **_pos** [3]
- float **_EDep**
- float **_time**
- **EVENT::MCParticle** ∗ **_particle**
- float **_p** [3]
  <– fixme: ROOT cannot handle this pointer !!????
- float **_pathLength**

Additional Inherited Members

7.139.1 Detailed Description

Implementation of SimTrackerHit.

See Also

SimTrackerHit

Author

gaede

Version

Mar 6, 2003

7.139.2 Member Function Documentation

7.139.2.1 int IMPL::SimTrackerHitImpl::getCellID1 ( ) const [virtual]

Same name as in CalorimeterHit, even though there are no 'cells' in this case 0 if information is not stored - check the flag word (bit THBIT_ID1) of the collection.

Default is to store only cellid0.

Implements **EVENT::SimTrackerHit** (p. 214).

7.139.2.2 float IMPL::SimTrackerHitImpl::getdEdx ( ) const [virtual]

Returns the dE/dx of the hit.

See Also

Deprecated **getEDep()** (p. 216)

Implements **EVENT::SimTrackerHit** (p. 215).

References getEDep(), and UTIL::LCWarning::printWarning().

7.139.2.3 EVENT::MCParticle ∗ IMPL::SimTrackerHitImpl::getMCParticle ( ) const [virtual]

Returns the MC particle that caused the hit.

See Also

MCParticle

Implements **EVENT::SimTrackerHit** (p. 215).

7.139.2.4 float IMPL::SimTrackerHitImpl::getPathLength ( ) const [virtual]

The path length of the particle in the sensitive material that resulted in this hit.

This is only stored together with momentum, i.e. if LCIO::THBIT_MOMENTUM is set.

Implements **EVENT::SimTrackerHit** (p. 215).

7.139.2.5 float IMPL::SimTrackerHitImpl::getTime ( ) const [virtual]

Returns the time of the hit.

TO DO needs definition.

Implements **EVENT::SimTrackerHit** (p. 215).

7.139.2.6 void IMPL::SimTrackerHitImpl::setCellID1 (int *id1*)

Sets the second cell id; Only store if the flag word (bit THBIT_ID1) of the collection is set.

Default is to store only cellid0.

7.139.2.7 void IMPL::SimTrackerHitImpl::setdEdx (float *dEdX*)

Sets dE/dx.

See Also

Deprecated **setEDep()** (p. 217)

References UTIL::LCWarning::printWarning(), and setEDep().

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

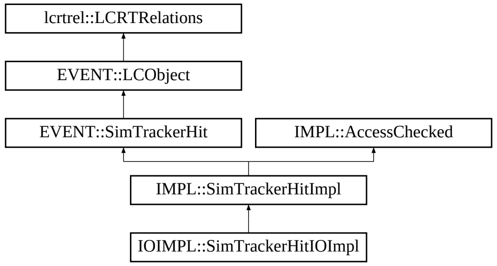
- SimTrackerHitImpl.h
- SimTrackerHitImpl.cc

7.140 IOIMPL::SimTrackerHitIOImpl Class Reference

Adding stuff needed for io (friend declarations, etc.)

`#include <IOIMPL/SimTrackerHitIOImpl.h>`

Inheritance diagram for IOIMPL::SimTrackerHitIOImpl:



Friends

- class **SIO::SIOSimTrackHitHandler**

Additional Inherited Members

7.140.1 Detailed Description

Adding stuff needed for io (friend declarations, etc.)

Author

gaede

Version

Mar 12, 2003

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

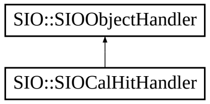
- SimTrackerHitIOImpl.h

7.141 SIO::SIOCalHitHandler Class Reference

Implementation of **SIOObjectHandler** (p. 229) to handle **IO** (p. 27) of CalorimeterHits.

`#include <SIO/SIOCalHitHandler.h>`

Inheritance diagram for SIO::SIOCalHitHandler:



Public Member Functions

- virtual unsigned int **read** (SIO_stream ∗stream, **EVENT::LCObject** ∗∗objP)
  Reads lcio calorimeter hit objects from an ***SIO*** *(p. 30) stream.*
- virtual unsigned int **write** (SIO_stream ∗stream, const **EVENT::LCObject** ∗obj)
  Writes lcio calorimeter hit objects to an ***SIO*** *(p. 30) stream.*

Additional Inherited Members

7.141.1 Detailed Description

Implementation of **SIOObjectHandler** (p. 229) to handle **IO** (p. 27) of CalorimeterHits.

Author

gaede

Version

Id:

SIOCalHitHandler.h (p. **??**),v 1.7 2005-04-15 08:37:42 gaede Exp

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

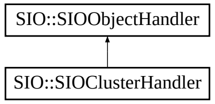
- SIOCalHitHandler.h
- SIOCalHitHandler.cc

7.142 SIO::SIOClusterHandler Class Reference

Implementation of **SIOObjectHandler** (p. 229) to handle **IO** (p. 27) of Clusters.

`#include <SIO/SIOClusterHandler.h>`

Inheritance diagram for SIO::SIOClusterHandler:



Public Member Functions

- virtual unsigned int **read** (SIO_stream ∗stream, **EVENT::LCObject** ∗∗objP)
  Reads lcio clusterer hit objects from an ***SIO*** *(p. 30) stream.*
- virtual unsigned int **write** (SIO_stream ∗stream, const **EVENT::LCObject** ∗obj)
  Writes lcio objects to an ***SIO*** *(p. 30) stream.*

Additional Inherited Members

7.142.1 Detailed Description

Implementation of **SIOObjectHandler** (p. 229) to handle **IO** (p. 27) of Clusters.

Author

gaede

Version

Id:

SIOClusterHandler.h (p. **??**),v 1.4 2005-04-15 08:37:42 gaede Exp

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

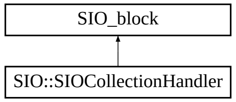
- SIOClusterHandler.h
- SIOClusterHandler.cc

7.143 SIO::SIOCollectionHandler Class Reference

Handler for LCCollection/LCCollectionIOVec objects for **SIO** (p. 30).

`#include <SIO/SIOCollectionHandler.h>`

Inheritance diagram for SIO::SIOCollectionHandler:



Public Member Functions

- **SIOCollectionHandler** (const std::string &name, const std::string &type, **IOIMPL::LCEventIOImpl** ∗∗anEvtP=0) throw (EVENT::Exception)
  The default constructor needs the name, the type and a pointer the address of the collection.
- const std::string & **getTypeName** () const
- virtual unsigned int **xfer** (SIO_stream ∗, SIO_operation, unsigned int)
- virtual unsigned int **version** ()
- void **setCollection** (const **EVENT::LCCollection** ∗col)
- void **setEvent** (**IOIMPL::LCEventIOImpl** ∗∗anEvtP)

7.143.1 Detailed Description

Handler for LCCollection/LCCollectionIOVec objects for **SIO** (p. 30).

Author

gaede

Version

Id:

SIOCollectionHandler.h (p. **??**),v 1.8 2005-04-15 08:37:42 gaede Exp

7.143.2 Constructor & Destructor Documentation

7.143.2.1 SIO::SIOCollectionHandler::SIOCollectionHandler (const std::string & *name*, const std::string & *type*, IOIMPL::LCEventIOImpl ∗∗ *anEvtP* = 0) throw EVENT::Exception)

The default constructor needs the name, the type and a pointer the address of the collection.

Throws an exception if handler for given type doesn't exist.

Exceptions

Exception	

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

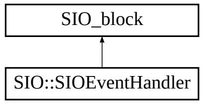
- SIOCollectionHandler.h
- SIOCollectionHandler.cc

7.144 SIO::SIOEventHandler Class Reference

Handler for LCEvent/LCEventIOImpl objects.

`#include <SIO/SIOEventHandler.h>`

Inheritance diagram for SIO::SIOEventHandler:



Public Member Functions

- **SIOEventHandler** (const std::string &name)
- **SIOEventHandler** (const std::string &name, **IOIMPL::LCEventIOImpl** ∗∗evtP)
- virtual unsigned int **xfer** (SIO_stream ∗, SIO_operation, unsigned int)
- virtual unsigned int **version** ()
- void **setEvent** (const **EVENT::LCEvent** ∗evt)
- void **setEventPtr** (**IOIMPL::LCEventIOImpl** ∗∗evtP)
- void **setReadCollectionNames** (const std::vector< std::string > &colnames)

7.144.1 Detailed Description

Handler for LCEvent/LCEventIOImpl objects.

Author

gaede

Version

Id:

SIOEventHandler.h (p. **??**),v 1.8 2005-04-15 08:37:42 gaede Exp

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

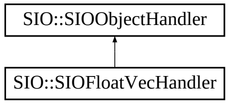
- SIOEventHandler.h
- SIOEventHandler.cc

7.145 SIO::SIOFloatVecHandler Class Reference

Implementation of **SIOObjectHandler** (p. 229) to handle **IO** (p. 27) of FloatVecs.

`#include <SIO/SIOFloatVecHandler.h>`

Inheritance diagram for SIO::SIOFloatVecHandler:



Public Member Functions

- virtual unsigned int **read** (SIO_stream ∗stream, **EVENT::LCObject** ∗∗objP)
  *Reads lcio FloatVecs objects from an **SIO** (p. 30) stream.*
- virtual unsigned int **write** (SIO_stream ∗stream, const **EVENT::LCObject** ∗obj)
  *Writes lcio FloatVecs to an **SIO** (p. 30) stream.*

Additional Inherited Members

7.145.1 Detailed Description

Implementation of **SIOObjectHandler** (p. 229) to handle **IO** (p. 27) of FloatVecs.

Author

gaede

Version

Id:

SIOFloatVecHandler.h (p. **??**),v 1.6 2005-04-15 08:37:42 gaede Exp

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

- SIOFloatVecHandler.h
- SIOFloatVecHandler.cc

7.146 SIO::SIOHandlerMgr Class Reference

Singleton that holds instances of all types of SIOObjectHandlers.

`#include <SIO/SIOHandlerMgr.h>`

Public Member Functions

- **SIOObjectHandler** ∗ **getHandler** (const std::string &type)
  Returns the instance of the handler for the given type - null if it doesn't exist.
- bool **registerHandler** (const std::string &type, **SIOObjectHandler** ∗handler)
  Registers an instance of a handler for the given type.

Static Public Member Functions

- static **SIOHandlerMgr** ∗ **instance** ()
  Returns the instance of this class.

7.146.1 Detailed Description

Singleton that holds instances of all types of SIOObjectHandlers.

New types need to have their handler classes registered here.

Author

gaede

Version

Id:

SIOHandlerMgr.h (p. **??**),v 1.5 2005-04-15 08:37:42 gaede Exp

7.146.2 Member Function Documentation

7.146.2.1 bool SIO::SIOHandlerMgr::registerHandler (const std::string & *type*, SIOObjectHandler ∗ *handler*)

Registers an instance of a handler for the given type.

Fails if type name allready exists.

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

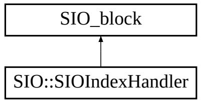
- SIOHandlerMgr.h
- SIOHandlerMgr.cc

7.147 SIO::SIOIndexHandler Class Reference

Handler for LCIOrandomAccess and LCIOIndex objects/blocks.

`#include <SIO/SIOIndexHandler.h>`

Inheritance diagram for SIO::SIOIndexHandler:



Public Member Functions

- **SIOIndexHandler** (const std::string &name, **LCIORandomAccessMgr** ∗raMgr)

  C'tor.
- virtual unsigned int **xfer** (SIO_stream ∗, SIO_operation, unsigned int)
- virtual unsigned int **version** ()

7.147.1 Detailed Description

Handler for LCIOrandomAccess and LCIOIndex objects/blocks.

Author

gaede

Version

Id:

SIOIndexHandler.h (p. **??**),v 1.2 2010-06-22 13:49:54 gaede Exp

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

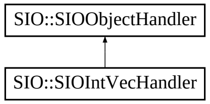
- SIOIndexHandler.h
- SIOIndexHandler.cc

7.148 SIO::SIOIntVecHandler Class Reference

Implementation of **SIOObjectHandler** (p. 229) to handle **IO** (p. 27) of IntVecs.

`#include <SIO/SIOIntVecHandler.h>`

Inheritance diagram for SIO::SIOIntVecHandler:



Public Member Functions

- virtual unsigned int **read** (SIO_stream ∗stream, **EVENT::LCObject** ∗∗objP)

  *Reads IntVec objects from an **SIO** (p. 30) stream.*
- virtual unsigned int **write** (SIO_stream ∗stream, const **EVENT::LCObject** ∗obj)

  *Writes lcio IntVec objects to an **SIO** (p. 30) stream.*

Additional Inherited Members

7.148.1 Detailed Description

Implementation of **SIOObjectHandler** (p. 229) to handle **IO** (p. 27) of IntVecs.

Author

gaede

Version

Id:

SIOIntVecHandler.h (p. **??**),v 1.4 2005-04-15 08:37:42 gaede Exp

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

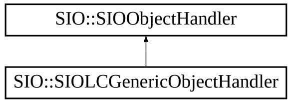
- SIOIntVecHandler.h
- SIOIntVecHandler.cc

7.149 SIO::SIOLCGenericObjectHandler Class Reference

Implementation of **SIOObjectHandler** (p. 229) to handle **IO** (p. 27) of LCGenericObjects.

`#include <SIO/SIOLCGenericObjectHandler.h>`

Inheritance diagram for SIO::SIOLCGenericObjectHandler:



Public Member Functions

- virtual unsigned int **read** (SIO_stream ∗stream, **EVENT::LCObject** ∗∗objP)

  *Reads lcio LCGenericObject objects from an **SIO** (p. 30) stream.*
- virtual unsigned int **write** (SIO_stream ∗stream, const **EVENT::LCObject** ∗obj)

  *Writes lcio LCGenericObject objects to an **SIO** (p. 30) stream.*
- virtual unsigned int **init** (SIO_stream ∗stream, SIO_operation op, **EVENT::LCCollection** ∗col, unsigned int vers)

  Initialize the handler and/or the collection.

Protected Attributes

- int **_nInt**
- int **_nFloat**
- int **_nDouble**
- bool **_isFixedSize**

7.149.1 Detailed Description

Implementation of **SIOObjectHandler** (p. 229) to handle **IO** (p. 27) of LCGenericObjects.

Author

gaede

Version

Id:

SIOLCGenericObjectHandler.h (p. **??**),v 1.3 2005-04-15 08:37:42 gaede Exp

7.149.2 Member Function Documentation

7.149.2.1 unsigned int SIO::SIOLCGenericObjectHandler::init (SIO_stream ∗ *stream,* SIO_operation *op,* EVENT::LCCollection ∗ *col,* unsigned int *vers*) [virtual]

Initialize the handler and/or the collection.

Read/write the flag and the parameters. Overwrite for classes that need specific processing.

Reimplemented from **SIO::SIOObjectHandler** (p. 231).

References IMPL::LCFlagImpl::bitSet(), EVENT::LCCollection::getElementAt(), EVENT::LCCollection::getFlag(), EVENT::LCCollection::getNumberOfElements(), EVENT::LCCollection::getParameters(), EVENT::LCParameters::getStringVal(), EVENT::LCGenericObject::isFixedSize(), EVENT::LCCollection::parameters(), EVENT::LCCollection::setFlag(), and EVENT::LCParameters::setValue().

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

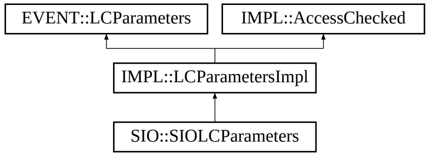
- SIOLCGenericObjectHandler.h
- SIOLCGenericObjectHandler.cc

7.150 SIO::SIOLCParameters Class Reference

IO (p. 27) of LCParameters.

`#include <SIO/SIOLCParameters.h>`

Inheritance diagram for SIO::SIOLCParameters:



Static Public Member Functions

- static unsigned int **read** (SIO_stream ∗stream, **LCParameters** &params, unsigned int vers)

  *Reads objects from an **SIO** (p. 30) stream.*
- static unsigned int **write** (SIO_stream ∗stream, const **LCParameters** &params)

  *Writes lcio objects to an **SIO** (p. 30) stream.*

Additional Inherited Members

7.150.1 Detailed Description

IO (p. 27) of LCParameters.

Author

gaede

Version

Id:

SIOLCParameters.h (p. **??**),v 1.4 2005-04-15 08:37:42 gaede Exp

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

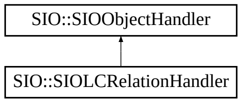
- SIOLCParameters.h
- SIOLCParameters.cc

7.151 SIO::SIOLCRelationHandler Class Reference

Implementation of **SIOObjectHandler** (p. 229) to handle **IO** (p. 27) of CalorimeterHits.

`#include <SIO/SIOLCRelationHandler.h>`

Inheritance diagram for SIO::SIOLCRelationHandler:



Public Member Functions

- virtual unsigned int **read** (SIO_stream ∗stream, **EVENT::LCObject** ∗∗objP)

  *Reads lcio relation objects from an **SIO** (p. 30) stream.*
- virtual unsigned int **write** (SIO_stream ∗stream, const **EVENT::LCObject** ∗obj)

  *Writes lcio objects to an **SIO** (p. 30) stream.*

Additional Inherited Members

7.151.1 Detailed Description

Implementation of **SIOObjectHandler** (p. 229) to handle **IO** (p. 27) of CalorimeterHits.

Author

gaede

Version

Id:

SIOLCRelationHandler.h (p. **??**),v 1.2 2005-04-15 08:37:42 gaede Exp

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

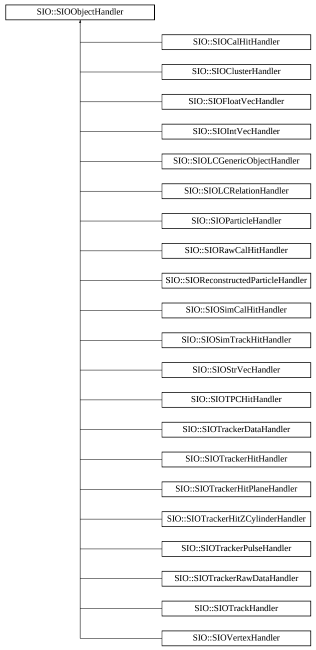
- SIOLCRelationHandler.h
- SIOLCRelationHandler.cc

7.152 SIO::SIOObjectHandler Class Reference

Interface for all lcio object SIO-handlers, has to be implemented for all event entities (hits, tracks, clusters,...).

`#include <SIO/SIOObjectHandler.h>`

Inheritance diagram for SIO::SIOObjectHandler:



Public Member Functions

- virtual unsigned int **read** (SIO_stream ∗stream, **EVENT::LCObject** ∗∗objP)=0

  *Reads lcio objects from an **SIO** (p. 30) stream.*
- virtual unsigned int **write** (SIO_stream ∗stream, const **EVENT::LCObject** ∗obj)=0

  *Writes lcio objects to an **SIO** (p. 30) stream.*
- virtual unsigned int **init** (SIO_stream ∗stream, SIO_operation op, **EVENT::LCCollection** ∗col, unsigned int vers)

  Initialize the handler and/or the collection.
- virtual unsigned int **readBase** (SIO_stream ∗stream, **EVENT::LCObject** ∗∗objP)

  *Calls **read()** (p. 230) if the collection is not a subset otherwise only reads the pointers.*
- virtual unsigned int **writeBase** (SIO_stream ∗stream, const **EVENT::LCObject** ∗obj)

  *Calls **write()** (p. 230) if the collection is not a subset otherwise only writes the pointers.*

Protected Attributes

- unsigned int **_flag**
- unsigned int **_vers**

7.152.1 Detailed Description

Interface for all lcio object SIO-handlers, has to be implemented for all event entities (hits, tracks, clusters,...).

Author

gaede

Version

Id:

SIOObjectHandler.h (p. **??**),v 1.10 2008-11-27 16:53:35 engels Exp

7.152.2 Member Function Documentation

7.152.2.1 unsigned int SIO::SIOObjectHandler::init (SIO_stream ∗ *stream*, SIO_operation *op*, EVENT::LCCollection ∗ *col*, unsigned int *vers*) [virtual]

Initialize the handler and/or the collection.

Read/write the flag and the parameters. Overwrite for classes that need specific processing.

Reimplemented in **SIO::SIOLCGenericObjectHandler** (p. 227), and **SIO::SIOVertexHandler** (p. 253).

References EVENT::LCCollection::getFlag(), EVENT::LCCollection::getParameters(), EVENT::LCCollection::parameters(), and EVENT::LCCollection::setFlag().

7.152.2.2 unsigned int SIO::SIOObjectHandler::readBase (SIO_stream ∗ *stream*, EVENT::LCObject ∗∗ *objP*) [virtual]

Calls **read()** (p. 230) if the collection is not a subset otherwise only reads the pointers.

7.152.2.3 unsigned int SIO::SIOObjectHandler::writeBase (SIO_stream ∗ *stream*, const EVENT::LCObject ∗ *obj*) [virtual]

Calls **write()** (p. 230) if the collection is not a subset otherwise only writes the pointers.

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

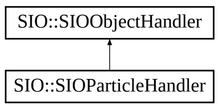
- SIOObjectHandler.h
- SIOObjectHandler.cc

7.153 SIO::SIOParticleHandler Class Reference

Implementation of **SIOObjectHandler** (p. 229) to handle **IO** (p. 27) of MCParticles.

`#include <SIO/SIOParticleHandler.h>`

Inheritance diagram for SIO::SIOParticleHandler:



Public Member Functions

- virtual unsigned int **read** (SIO_stream ∗stream, **EVENT::LCObject** ∗∗objP)
  *Reads lcio MCParticle objects from an **SIO** (p. 30) stream.*
- virtual unsigned int **write** (SIO_stream ∗stream, const **EVENT::LCObject** ∗obj)
  *Writes lcio MCParticle objects to an **SIO** (p. 30) stream.*

Static Public Member Functions

- static void **restoreParentDaughterRelations** (**EVENT::LCEvent** ∗evt)

Additional Inherited Members

7.153.1 Detailed Description

Implementation of **SIOObjectHandler** (p. 229) to handle **IO** (p. 27) of MCParticles.

Author

gaede

Version

Id:

SIOParticleHandler.h (p. **??**),v 1.8 2005-04-15 08:37:42 gaede Exp

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

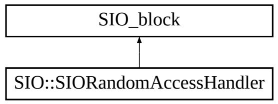
- SIOParticleHandler.h
- SIOParticleHandler.cc

7.154 SIO::SIORandomAccessHandler Class Reference

Handler for LCIOrandomAccess and LCIOIndex objects/blocks.

`#include <SIO/SIORandomAccessHandler.h>`

Inheritance diagram for SIO::SIORandomAccessHandler:



Public Member Functions

- **SIORandomAccessHandler** (const std::string &name, **LCIORandomAccessMgr** ∗raMgr)
  C'tor.
- virtual unsigned int **xfer** (SIO_stream ∗, SIO_operation, unsigned int)
- virtual unsigned int **version** ()

7.154.1 Detailed Description

Handler for LCIOrandomAccess and LCIOIndex objects/blocks.

Author

gaede

Version

Id:

SIORandomAccessHandler.h (p. **??**),v 1.2 2010-06-22 13:49:54 gaede Exp

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

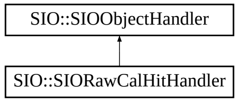
- SIORandomAccessHandler.h
- SIORandomAccessHandler.cc

7.155 SIO::SIORawCalHitHandler Class Reference

Implementation of **SIOObjectHandler** (p. 229) to handle **IO** (p. 27) of RawCalorimeterHits.

`#include <SIO/SIORawCalHitHandler.h>`

Inheritance diagram for SIO::SIORawCalHitHandler:



Public Member Functions

- virtual unsigned int **read** (SIO_stream ∗stream, **EVENT::LCObject** ∗∗objP)
  *Reads lcio raw calorimeter hit objects from an **SIO** (p. 30) stream.*
- virtual unsigned int **write** (SIO_stream ∗stream, const **EVENT::LCObject** ∗obj)
  *Writes lcio raw calorimeter hit objects to an **SIO** (p. 30) stream.*

Additional Inherited Members

7.155.1 Detailed Description

Implementation of **SIOObjectHandler** (p. 229) to handle **IO** (p. 27) of RawCalorimeterHits.

Author

gaede

Version

Id:

SIORawCalHitHandler.h (p. **??**),v 1.2 2005-04-15 08:37:42 gaede Exp

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

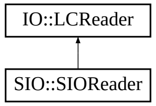
- SIORawCalHitHandler.h
- SIORawCalHitHandler.cc

7.156 SIO::SIOReader Class Reference

Concrete implementation of LCWriter using **SIO** (p. 30).

`#include <SIO/SIOReader.h>`

Inheritance diagram for SIO::SIOReader:



Public Member Functions

- **SIOReader** (int lcReaderFlag=0)
  Default constructor.
- virtual void **open** (const std::vector< std::string > &filenames) throw (IO::IOException, std::exception)
  Opens a list of files for reading (read-only).
- virtual void **open** (const std::string &filename) throw (IO::IOException, std::exception)
  Opens a file for reading (read-only).
- virtual **EVENT::LCRunHeader** ∗ **readNextRunHeader** () throw (IO::IOException, std::exception)
  Reads the next run header from the file.
- virtual **EVENT::LCRunHeader** ∗ **readNextRunHeader** (int accessMode) throw (IO::IOException, std::exception)
  *Same as **readNextRunHeader()** (p. 239) but allows to set the access mode LCIO::READ_ONLY (default) or LCIO::Update.*
- virtual **EVENT::LCEvent** ∗ **readNextEvent** () throw (IO::IOException, std::exception)
  Reads the next event from the file.
- virtual **EVENT::LCEvent** ∗ **readNextEvent** (int accessMode) throw (IO::IOException, std::exception)
  *Same as **readNextRunHeader()** (p. 239) but allows to set the access mode LCIO::READ_ONLY (default) or LCIO::Update.*
- virtual int **getNumberOfEvents** () throw (IO::IOException, std::exception)
  Return the number of events in the file - the file has to be open.
- virtual int **getNumberOfRuns** () throw (IO::IOException, std::exception)
  Return the number of runs (run headers) in the file - the file has to be open.
- virtual void **getRuns** (**EVENT::IntVec** &runs)
  Return the run numbers of the runs (run headers) in the file - the file has to be open.
- virtual void **getEvents** (**EVENT::IntVec** &events)
  Return the run and event numbers of the events in the file - the file has to be open.

- virtual void **skipNEvents** (int n) throw (IO::IOException, std::exception)

  Skips the next n events from the current position.
- virtual void **setReadCollectionNames** (const std::vector< std::string > &colnames)

  Limit the collection names that are going to be read to the subset given in the vector - all other collection will be ignored.
- virtual **EVENT::LCRunHeader** ∗ **readRunHeader** (int runNumber) throw (IO::IOException , std::exception)

  Reads the specified runHeader from file.
- virtual **EVENT::LCRunHeader** ∗ **readRunHeader** (int runNumber, int accessMode) throw (IO::IOException , std::exception)

  *Same as LCEvent∗ **readRunHeader(int runNumber)** (p. 239) allowing to set the access mode LCIO::READ_ONLY (default) or LCIO::Update.*
- virtual **EVENT::LCEvent** ∗ **readEvent** (int runNumber, int evtNumber) throw (IO::IOException, std::exception)

  Reads the specified event from file.
- virtual **EVENT::LCEvent** ∗ **readEvent** (int runNumber, int evtNumber, int accessMode) throw (IO::IOException, std::exception)

  Same as LCEvent∗ readEvent(int runNumber, int evtNumber allowing to set the access mode LCIO::READ_ONLY (default) or LCIO::Update.
- virtual void **close** () throw (IO::IOException, std::exception)

  Closes the output file/stream etc.
- virtual void **registerLCEventListener** (**IO::LCEventListener** ∗ls)

  Registers a listener for reading LCEvents from a stream.
- virtual void **removeLCEventListener** (**IO::LCEventListener** ∗ls)

  Remove a listener for reading LCEvents from a stream.
- virtual void **registerLCRunListener** (**IO::LCRunListener** ∗ls)

  Registers a listener for reading LCEventsLCRunHeaders from a stream.
- virtual void **removeLCRunListener** (**IO::LCRunListener** ∗ls)

  Remove a listener for reading LCRunHeaders from a stream.
- virtual void **readStream** () throw (IO::IOException, std::exception)

  Reads the input stream and notifies registered listeners according to the object type found in the stream.
- virtual void **readStream** (int maxRecord) throw (IO::IOException, std::exception)

  Reads maxRecord from the input stream and notifies registered listeners according to the object type found in the stream.

Protected Member Functions

- void **setUpHandlers** ()
- void **readRecord** () throw (IO::IOException , IO::EndOfDataException , std::exception)
- void **postProcessEvent** ()
- void **getEventMap** ()
- void **recreateEventMap** ()

Protected Attributes

- SIO_record ∗ **_dummyRecord**
- SIO_stream ∗ **_stream**
- **SIORunHeaderHandler** ∗ **_runHandler**
- **SIOEventHandler** ∗ **_evtHandler**

Additional Inherited Members

7.156.1 Detailed Description

Concrete implementation of LCWriter using **SIO** (p. 30).

Author

gaede

Version

Id:

SIOReader.h (p. **??**),v 1.29 2011-03-03 16:00:12 gaede Exp

7.156.2 Member Function Documentation

7.156.2.1 void SIO::SIOReader::close ( ) throw IO::IOException, std::exception) [virtual]

Closes the output file/stream etc.

Exceptions

IOException	

Implements **IO::LCReader** (p. 132).

References SIO::LCIORandomAccessMgr::clear().

7.156.2.2 void SIO::SIOReader::getEvents (EVENT::IntVec & *events*) [virtual]

Return the run and event numbers of the events in the file - the file has to be open.

In case several input files are specified in the **open()** (p. 237) method - the run and event numbers of the events in the file that is currently open are returned. The size of the vector events will be twice the number of events, where i-th run number will be in events[2∗i] and the i-th event number in events[2∗i+].

Implements **IO::LCReader** (p. 133).

References SIO::LCIORandomAccessMgr::getEventMap(), getNumberOfEvents(), and getNumberOfRuns().

7.156.2.3 int SIO::SIOReader::getNumberOfEvents ( ) throw IO::IOException, std::exception) [virtual]

Return the number of events in the file - the file has to be open.

In case several input files are specified in the **open()** (p. 237) method - the number of events in the file that is currently open is returned.

Implements **IO::LCReader** (p. 133).

References SIO::LCIORandomAccessMgr::getEventMap().

Referenced by getEvents().

7.156.2.4 int SIO::SIOReader::getNumberOfRuns ( ) throw IO::IOException, std::exception) [virtual]

Return the number of runs (run headers) in the file - the file has to be open.

In case several input files are specified in the **open()** (p. 237) method - the number of runs (run headers) in the file that is currently open is returned.

Implements **IO::LCReader** (p. 133).

References SIO::LCIORandomAccessMgr::getEventMap().

Referenced by getEvents(), and getRuns().

7.156.2.5 void SIO::SIOReader::getRuns (EVENT::IntVec & *runs*) [virtual]

Return the run numbers of the runs (run headers) in the file - the file has to be open.

In case several input files are specified in the **open()** (p. 237) method - the run numbers of the runs (run headers) in the file that is currently open is returned.

Implements **IO::LCReader** (p. 133).

References SIO::LCIORandomAccessMgr::getEventMap(), and getNumberOfRuns().

7.156.2.6 void SIO::SIOReader::open (const std::vector< std::string > & *filenames*) throw IO::IOException, std::exception) [virtual]

Opens a list of files for reading (read-only).

All subsequent read operations will operate on the list, i.e. if an EOF is encountered the next file in the list will be opened and read transparently to the user.

Exceptions

IOException	

Implements **IO::LCReader** (p. 133).

7.156.2.7 void SIO::SIOReader::open (const std::string & *filename*) throw IO::IOException, std::exception) [virtual]

Opens a file for reading (read-only).

Exceptions

IOException	

Implements **IO::LCReader** (p. 133).

References SIO::LCSIO::getValidSIOName().

7.156.2.8 EVENT::LCEvent ∗ SIO::SIOReader::readEvent (int *runNumber,* int *evtNumber*) throw IO::IOException, std::exception) [virtual]

Reads the specified event from file.

Returns NULL if the specified event hasn't been found in the file.

Exceptions

IOException	

Implements **IO::LCReader** (p. 134).

7.156.2.9 EVENT::LCEvent ∗ SIO::SIOReader::readEvent (int *runNumber,* int *evtNumber,* int *accessMode*) throw IO::IOException, std::exception) [virtual]

Same as LCEvent∗ readEvent(int runNumber, int evtNumber allowing to set the access mode LCIO::READ_ONLY (default) or LCIO::Update.

Exceptions

IOException	

Implements **IO::LCReader** (p. 134).

7.156.2.10 LCEvent ∗ SIO::SIOReader::readNextEvent ( ) throw IO::IOException, std::exception) [virtual]

Reads the next event from the file.

Exceptions

IOException	

Implements **IO::LCReader** (p. 134).

7.156.2.11 LCEvent ∗ SIO::SIOReader::readNextEvent (int *accessMode*) throw IO::IOException, std::exception) [virtual]

Same as **readNextRunHeader()** (p. 239) but allows to set the access mode LCIO::READ_ONLY (default) or LCIO-::Update.

Exceptions

IOException	

Implements **IO::LCReader** (p. 134).

7.156.2.12 LCRunHeader ∗ SIO::SIOReader::readNextRunHeader ( ) throw IO::IOException, std::exception) [virtual]

Reads the next run header from the file.

Exceptions

IOException	

Implements **IO::LCReader** (p. 134).

7.156.2.13 LCRunHeader ∗ SIO::SIOReader::readNextRunHeader (int *accessMode*) throw IO::IOException, std::exception) [virtual]

Same as **readNextRunHeader()** (p. 239) but allows to set the access mode LCIO::READ_ONLY (default) or LCIO-::Update.

Exceptions

IOException	

Implements **IO::LCReader** (p. 135).

7.156.2.14 EVENT::LCRunHeader ∗ SIO::SIOReader::readRunHeader (int *runNumber*) throw IO::IOException , std::exception) [virtual]

Reads the specified runHeader from file.

Returns NULL if the specified runHeader hasn't been found in the file.

Exceptions

IOException	

Implements **IO::LCReader** (p. 135).

7.156.2.15 EVENT::LCRunHeader ∗ SIO::SIOReader::readRunHeader (int *runNumber,* int *accessMode*) throw IO::IOException , std::exception) [virtual]

Same as LCEvent∗ **readRunHeader(int runNumber)** (p. 239) allowing to set the access mode LCIO::READ_ONLY (default) or LCIO::Update.

Exceptions

IOException	

Implements **IO::LCReader** (p. 135).

7.156.2.16 void SIO::SIOReader::readStream ( ) throw IO::IOException, std::exception) [virtual]

Reads the input stream and notifies registered listeners according to the object type found in the stream.

Exceptions

IOException	
EndOfException	

Implements **IO::LCReader** (p. 135).

7.156.2.17 void SIO::SIOReader::readStream (int *maxRecord*) throw IO::IOException, std::exception) [virtual]

Reads maxRecord from the input stream and notifies registered listeners according to the object type found in the stream.

Throws EndOfException if less than maxRecord records are found in the stream.

Exceptions

IOException	
EndOfException	

Implements **IO::LCReader** (p. 135).

7.156.2.18 void SIO::SIOReader::setReadCollectionNames (const std::vector< std::string > & *colnames*) [virtual]

Limit the collection names that are going to be read to the subset given in the vector - all other collection will be ignored.

This might improve the reading performance considerably in cases where only a small subset of the collections in the event is needed.

Implements **IO::LCReader** (p. 136).

7.156.2.19 void SIO::SIOReader::skipNEvents (int *n*) throw IO::IOException, std::exception) [virtual]

Skips the next n events from the current position.

In fact simply reads the next n event headers so that the next event read is the (n+1)-th event.

Implements **IO::LCReader** (p. 136).

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

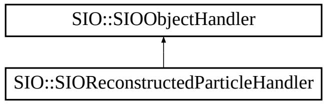
- SIOReader.h
- SIOReader.cc

7.157 SIO::SIOReconstructedParticleHandler Class Reference

Implementation of **SIOObjectHandler** (p. 229) to handle **IO** (p. 27) of ReconstructedParticles.

`#include <SIO/SIOReconstructedParticleHandler.h>`

Inheritance diagram for SIO::SIOReconstructedParticleHandler:



Public Member Functions

- virtual unsigned int **read** (SIO_stream ∗stream, **EVENT::LCObject** ∗∗objP)
  *Reads lcio ReconstructedParticle objects from an **SIO** (p. 30) stream.*
- virtual unsigned int **write** (SIO_stream ∗stream, const **EVENT::LCObject** ∗obj)
  *Writes lcio ReconstructedParticle objects to an **SIO** (p. 30) stream.*

Additional Inherited Members

7.157.1 Detailed Description

Implementation of **SIOObjectHandler** (p. 229) to handle **IO** (p. 27) of ReconstructedParticles.

Author

gaede

Version

Id:

SIOReconstructedParticleHandler.h (p. **??**),v 1.4 2005-04-15 08:37:43 gaede Exp

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

- SIOReconstructedParticleHandler.h
- SIOReconstructedParticleHandler.cc

7.158 SIO::SIORecords Class Reference

Manager class that holds instances of all known LCIO-SIO records.

`#include <SIO/LCSIO.h>`

Classes

- class **Unpack**
  *Helper class that sets the unpack flag of known **SIO** (p. 30) records as specified by the constructor.*

Public Types

- enum **ID** {
  Event =0, **Header**, **Run**, **Access**,
  Index, **NumberOfRecords** }

Public Member Functions

- SIO_record ∗ **operator[]** (size_t idx)
- void **setCompress** (bool flag=true)

Protected Member Functions

- void **add** (SIO_record ∗&rec, const char ∗name)

Protected Attributes

- SIO_record ∗ **_records** [NumberOfRecords]

Friends

- class **LCSIO**

7.158.1 Detailed Description

Manager class that holds instances of all known LCIO-SIO records.

Provides **Unpack** (p. 311) class for setting subsets of records to be unpacked. Every new record type needs to be added here in SIORecords::ID and **SIORecords::Unpack** (p. 311).

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

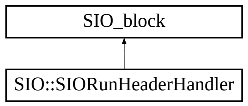
- LCSIO.h
- LCSIO.cc

7.159 SIO::SIORunHeaderHandler Class Reference

Handler for LCRunHeader/LCRunHeaderImpl objects.

`#include <SIO/SIORunHeaderHandler.h>`

Inheritance diagram for SIO::SIORunHeaderHandler:



Public Member Functions

- **SIORunHeaderHandler** (const std::string &name)
  C'tor for writing.
- **SIORunHeaderHandler** (const std::string &name, **IOIMPL::LCRunHeaderIOImpl** ∗∗rhP)
  C'tor for reading.
- virtual unsigned int **xfer** (SIO_stream ∗, SIO_operation, unsigned int)
- virtual unsigned int **version** ()
- void **setRunHeader** (const **EVENT::LCRunHeader** ∗hdr)
- void **setRunHeaderPtr** (**IOIMPL::LCRunHeaderIOImpl** ∗∗hdrP)

7.159.1 Detailed Description

Handler for LCRunHeader/LCRunHeaderImpl objects.

Author

gaede

Version

Id:

SIORunHeaderHandler.h (p. **??**),v 1.8 2005-04-15 08:37:43 gaede Exp

fg 20030609 using data interface for writing

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

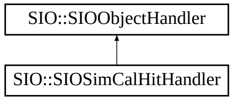
- SIORunHeaderHandler.h
- SIORunHeaderHandler.cc

7.160 SIO::SIOSimCalHitHandler Class Reference

Implementation of **SIOObjectHandler** (p. 229) to handle **IO** (p. 27) of SimCalorimeterHits.

`#include <SIO/SIOSimCalHitHandler.h>`

Inheritance diagram for SIO::SIOSimCalHitHandler:



Public Member Functions

- virtual unsigned int **read** (SIO_stream ∗stream, **EVENT::LCObject** ∗∗objP)
  *Reads lcio calorimeter hit objects from an **SIO** (p. 30) stream.*
- virtual unsigned int **write** (SIO_stream ∗stream, const **EVENT::LCObject** ∗obj)
  *Writes lcio objects to an **SIO** (p. 30) stream.*

Additional Inherited Members

7.160.1 Detailed Description

Implementation of **SIOObjectHandler** (p. 229) to handle **IO** (p. 27) of SimCalorimeterHits.

Author

gaede

Version

Id:

SIOSimCalHitHandler.h (p. **??**),v 1.6 2005-04-15 08:37:43 gaede Exp

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

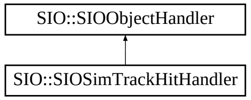
- SIOSimCalHitHandler.h
- SIOSimCalHitHandler.cc

7.161 SIO::SIOSimTrackHitHandler Class Reference

Implementation of **SIOObjectHandler** (p. 229) to handle **IO** (p. 27) of SimTrackerHits.

`#include <SIO/SIOSimTrackHitHandler.h>`

Inheritance diagram for SIO::SIOSimTrackHitHandler:



Public Member Functions

- virtual unsigned int **read** (SIO_stream ∗stream, **EVENT::LCObject** ∗∗objP)
  *Reads lcio tracker hit objects from an **SIO** (p. 30) stream.*
- virtual unsigned int **write** (SIO_stream ∗stream, const **EVENT::LCObject** ∗obj)
  *Writes lcio objects to an **SIO** (p. 30) stream.*

Additional Inherited Members

7.161.1 Detailed Description

Implementation of **SIOObjectHandler** (p. 229) to handle **IO** (p. 27) of SimTrackerHits.

Author

gaede

Version

Id:

SIOSimTrackHitHandler.h (p. **??**),v 1.5 2005-04-15 08:37:43 gaede Exp

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

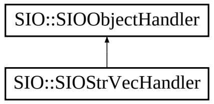
- SIOSimTrackHitHandler.h
- SIOSimTrackHitHandler.cc

7.162 SIO::SIOStrVecHandler Class Reference

Implementation of **SIOObjectHandler** (p. 229) to handle **IO** (p. 27) of StrVecs.

`#include <SIO/SIOStrVecHandler.h>`

Inheritance diagram for SIO::SIOStrVecHandler:



Public Member Functions

- virtual unsigned int **read** (SIO_stream ∗stream, **EVENT::LCObject** ∗∗objP)
  Reads objects from an ***SIO*** *(p. 30) stream.*
- virtual unsigned int **write** (SIO_stream ∗stream, const **EVENT::LCObject** ∗obj)
  Writes lcio objects to an ***SIO*** *(p. 30) stream.*

Additional Inherited Members

7.162.1 Detailed Description

Implementation of **SIOObjectHandler** (p. 229) to handle **IO** (p. 27) of StrVecs.

Author

vogt

Version

Id:

SIOStrVecHandler.h (p. **??**),v 1.3 2005-04-15 08:37:43 gaede Exp

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

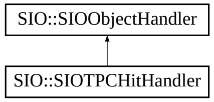
- SIOStrVecHandler.h
- SIOStrVecHandler.cc

7.163 SIO::SIOTPCHitHandler Class Reference

Implementation of **SIOObjectHandler** (p. 229) to handle **IO** (p. 27) of TPCHits.

`#include <SIO/SIOTPCHitHandler.h>`

Inheritance diagram for SIO::SIOTPCHitHandler:



Public Member Functions

- virtual unsigned int **read** (SIO_stream ∗stream, **EVENT::LCObject** ∗∗objP)
  Reads lcio TPC hit objects from an ***SIO*** *(p. 30) stream.*
- virtual unsigned int **write** (SIO_stream ∗stream, const **EVENT::LCObject** ∗obj)
  Writes lcio TPC hit objects to an ***SIO*** *(p. 30) stream.*

Additional Inherited Members

7.163.1 Detailed Description

Implementation of **SIOObjectHandler** (p. 229) to handle **IO** (p. 27) of TPCHits.

Author

gaede

Version

Id:

SIOTPCHitHandler.h (p. **??**),v 1.4 2005-04-15 08:37:43 gaede Exp

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

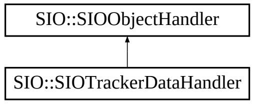
- SIOTPCHitHandler.h
- SIOTPCHitHandler.cc

7.164 SIO::SIOTrackerDataHandler Class Reference

Implementation of **SIOObjectHandler** (p. 229) to handle **IO** (p. 27) of TrackerDatas.

`#include <SIO/SIOTrackerDataHandler.h>`

Inheritance diagram for SIO::SIOTrackerDataHandler:



Public Member Functions

- virtual unsigned int **read** (SIO_stream ∗stream, **EVENT::LCObject** ∗∗objP)

  *Reads lcio TrackerData objects from an **SIO** (p. 30) stream.*
- virtual unsigned int **write** (SIO_stream ∗stream, const **EVENT::LCObject** ∗obj)

  *Writes TrackerData hit objects to an **SIO** (p. 30) stream.*

Additional Inherited Members

7.164.1 Detailed Description

Implementation of **SIOObjectHandler** (p. 229) to handle **IO** (p. 27) of TrackerDatas.

Author

gaede

Version

Id:

SIOTrackerDataHandler.h (p. **??**),v 1.1 2005-05-31 07:43:25 gaede Exp

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

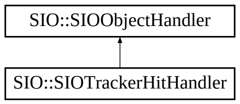
- SIOTrackerDataHandler.h
- SIOTrackerDataHandler.cc

7.165 SIO::SIOTrackerHitHandler Class Reference

Implementation of **SIOObjectHandler** (p. 229) to handle **IO** (p. 27) of SimTrackerHits.

`#include <SIO/SIOTrackerHitHandler.h>`

Inheritance diagram for SIO::SIOTrackerHitHandler:



Public Member Functions

- virtual unsigned int **read** (SIO_stream ∗stream, **EVENT::LCObject** ∗∗objP)

  *Reads lcio tracker hit objects from an **SIO** (p. 30) stream.*
- virtual unsigned int **write** (SIO_stream ∗stream, const **EVENT::LCObject** ∗obj)

  *Writes lcio tracker hit objects to an **SIO** (p. 30) stream.*

Additional Inherited Members

7.165.1 Detailed Description

Implementation of **SIOObjectHandler** (p. 229) to handle **IO** (p. 27) of SimTrackerHits.

Author

gaede

Version

Id:

SIOTrackerHitHandler.h (p. **??**),v 1.3 2005-04-15 08:37:43 gaede Exp

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

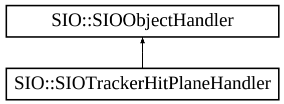
- SIOTrackerHitHandler.h
- SIOTrackerHitHandler.cc

7.166 SIO::SIOTrackerHitPlaneHandler Class Reference

Implementation of **SIOObjectHandler** (p. 229) to handle **IO** (p. 27) of TrackerHitPlane.

`#include <SIO/SIOTrackerHitPlaneHandler.h>`

Inheritance diagram for SIO::SIOTrackerHitPlaneHandler:



Public Member Functions

- virtual unsigned int **read** (SIO_stream ∗stream, **EVENT::LCObject** ∗∗objP)

  *Reads lcio object from an **SIO** (p. 30) stream.*
- virtual unsigned int **write** (SIO_stream ∗stream, const **EVENT::LCObject** ∗obj)

  *Writes lcio object to an **SIO** (p. 30) stream.*

Additional Inherited Members

7.166.1 Detailed Description

Implementation of **SIOObjectHandler** (p. 229) to handle **IO** (p. 27) of TrackerHitPlane.

Author

engels

Version

$Id:$

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

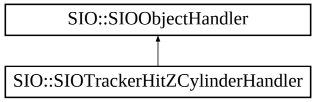
- SIOTrackerHitPlaneHandler.h
- SIOTrackerHitPlaneHandler.cc

7.167 SIO::SIOTrackerHitZCylinderHandler Class Reference

Implementation of **SIOObjectHandler** (p. 229) to handle **IO** (p. 27) of TrackerHitZCylinder.

`#include <SIO/SIOTrackerHitZCylinderHandler.h>`

Inheritance diagram for SIO::SIOTrackerHitZCylinderHandler:



Public Member Functions

- virtual unsigned int **read** (SIO_stream ∗stream, **EVENT::LCObject** ∗∗objP)

  Reads lcio object from an ***SIO*** *(p. 30) stream.*
- virtual unsigned int **write** (SIO_stream ∗stream, const **EVENT::LCObject** ∗obj)

  Writes lcio object to an ***SIO*** *(p. 30) stream.*

Additional Inherited Members

7.167.1 Detailed Description

Implementation of **SIOObjectHandler** (p. 229) to handle **IO** (p. 27) of TrackerHitZCylinder.

Author

engels

Version

$Id:$

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

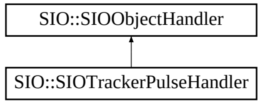
- SIOTrackerHitZCylinderHandler.h
- SIOTrackerHitZCylinderHandler.cc

7.168 SIO::SIOTrackerPulseHandler Class Reference

Implementation of **SIOObjectHandler** (p. 229) to handle **IO** (p. 27) of TrackerPulses.

`#include <SIO/SIOTrackerPulseHandler.h>`

Inheritance diagram for SIO::SIOTrackerPulseHandler:



Public Member Functions

- virtual unsigned int **read** (SIO_stream ∗stream, **EVENT::LCObject** ∗∗objP)

  Reads lcio TrackerPulse objects from an ***SIO*** *(p. 30) stream.*
- virtual unsigned int **write** (SIO_stream ∗stream, const **EVENT::LCObject** ∗obj)

  Writes TrackerPulse hit objects to an ***SIO*** *(p. 30) stream.*

Additional Inherited Members

7.168.1 Detailed Description

Implementation of **SIOObjectHandler** (p. 229) to handle **IO** (p. 27) of TrackerPulses.

Author

gaede

Version

Id:

SIOTrackerPulseHandler.h (p. **??**),v 1.1 2005-05-31 07:43:25 gaede Exp

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

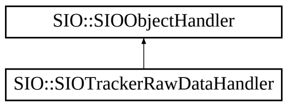
- SIOTrackerPulseHandler.h
- SIOTrackerPulseHandler.cc

7.169 SIO::SIOTrackerRawDataHandler Class Reference

Implementation of **SIOObjectHandler** (p. 229) to handle **IO** (p. 27) of TrackerRawDatas.

`#include <SIO/SIOTrackerRawDataHandler.h>`

Inheritance diagram for SIO::SIOTrackerRawDataHandler:



Public Member Functions

- virtual unsigned int **read** (SIO_stream ∗stream, **EVENT::LCObject** ∗∗objP)

  *Reads lcio TrackerRawData objects from an **SIO** (p. 30) stream.*
- virtual unsigned int **write** (SIO_stream ∗stream, const **EVENT::LCObject** ∗obj)

  *Writes TrackerRawData hit objects to an **SIO** (p. 30) stream.*

Additional Inherited Members

7.169.1 Detailed Description

Implementation of **SIOObjectHandler** (p. 229) to handle **IO** (p. 27) of TrackerRawDatas.

Author

gaede

Version

Id:

SIOTrackerRawDataHandler.h (p. **??**),v 1.1 2005-05-31 07:43:25 gaede Exp

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

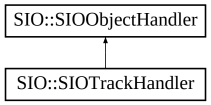
- SIOTrackerRawDataHandler.h
- SIOTrackerRawDataHandler.cc

7.170 SIO::SIOTrackHandler Class Reference

Implementation of **SIOObjectHandler** (p. 229) to handle **IO** (p. 27) of Tracks.

`#include <SIO/SIOTrackHandler.h>`

Inheritance diagram for SIO::SIOTrackHandler:



Public Member Functions

- virtual unsigned int **read** (SIO_stream ∗stream, **EVENT::LCObject** ∗∗objP)

  *Reads lcio track objects from an **SIO** (p. 30) stream.*
- virtual unsigned int **write** (SIO_stream ∗stream, const **EVENT::LCObject** ∗obj)

  *Writes lcio track objects to an **SIO** (p. 30) stream.*

Additional Inherited Members

7.170.1 Detailed Description

Implementation of **SIOObjectHandler** (p. 229) to handle **IO** (p. 27) of Tracks.

Author

gaede

Version

Id:

SIOTrackHandler.h (p. **??**),v 1.4 2005-04-15 08:37:43 gaede Exp

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

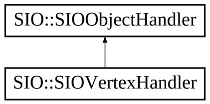
- SIOTrackHandler.h
- SIOTrackHandler.cc

7.171 SIO::SIOVertexHandler Class Reference

Implementation of **SIOObjectHandler** (p. 229) to handle **IO** (p. 27) of Vertexes.

`#include <SIO/SIOVertexHandler.h>`

Inheritance diagram for SIO::SIOVertexHandler:



Public Member Functions

- virtual unsigned int **init** (SIO_stream ∗stream, SIO_operation op, **EVENT::LCCollection** ∗col, unsigned int vers)

  Initialize the handler and/or the collection.
- virtual unsigned int **read** (SIO_stream ∗stream, **EVENT::LCObject** ∗∗objP)

  *Reads lcio Vertex objects from an **SIO** (p. 30) stream.*
- virtual unsigned int **write** (SIO_stream ∗stream, const **EVENT::LCObject** ∗obj)

  *Writes lcio Vertex objects to an **SIO** (p. 30) stream.*

Protected Attributes

- std::map< int, std::string > **imr**
- std::set< std::string > **_set**
- **EVENT::StringVec parameters**

7.171.1 Detailed Description

Implementation of **SIOObjectHandler** (p. 229) to handle **IO** (p. 27) of Vertexes.

Author

gaede, engels

Version

Id:

SIOVertexHandler.h (p. **??**),v 1.3 2006-12-06 19:09:18 hvogt Exp

7.171.2 Member Function Documentation

7.171.2.1 unsigned int SIO::SIOVertexHandler::init (SIO_stream ∗ *stream*, SIO_operation *op*, EVENT::LCCollection ∗ *col*, unsigned int *vers*) [virtual]

Initialize the handler and/or the collection.

Read/write the flag and the parameters. Overwrite for classes that need specific processing.

Reimplemented from **SIO::SIOObjectHandler** (p. 231).

References EVENT::Vertex::getAlgorithmType(), EVENT::LCCollection::getElementAt(), EVENT::LCCollection::getFlag(), EVENT::LCCollection::getNumberOfElements(), EVENT::LCCollection::getParameters(), EVENT::LCParameters::getStringVals(), EVENT::LCCollection::parameters(), EVENT::LCCollection::setFlag(), and EVENT::LCParameters::setValues().

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

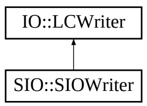
- SIOVertexHandler.h
- SIOVertexHandler.cc

7.172 SIO::SIOWriter Class Reference

Concrete implementation of LCWriter using **SIO** (p. 30).

`#include <SIO/SIOWriter.h>`

Inheritance diagram for SIO::SIOWriter:



Public Member Functions

- **SIOWriter** ()

  Default constructor.
- virtual ∼**SIOWriter** ()

  Destructor.

- virtual void **open** (const std::string &filename) throw (IO::IOException, std::exception)

  Opens a file for writing.
- virtual void **open** (const std::string &filename, int writeMode) throw (IO::IOException, std::exception)

  Opens a file for writing.
- virtual void **setCompressionLevel** (int level)

  *Set the compression level - needs to be called before **open()** (p. 256) otherwise call will have no effect.*
- virtual void **writeRunHeader** (const **EVENT::LCRunHeader** ∗hdr) throw (IO::IOException, std::exception)

  Writes the given run header to file.
- virtual void **writeEvent** (const **EVENT::LCEvent** ∗evt) throw (IO::IOException, std::exception)

  Writes the given event to file.
- virtual void **close** () throw (IO::IOException, std::exception)

  Closes the output file/stream etc.
- virtual void **flush** () throw (IO::IOException, std::exception)

  Flushes the output file/stream etc.

Protected Member Functions

- void **setUpHandlers** (const **EVENT::LCEvent** ∗evt)

  Sets up the handlers for writing the current event.
- void **getSIOFileName** (const std::string &filename, std::string &sioFilename)

  Creates a proper filename with extension 'slcio' in sioFilename.

Protected Attributes

- SIO_stream ∗ **_stream**
- int **_compressionLevel**
- **LCIORandomAccessMgr _raMgr**

Friends

- class **SIOReader**
- class **SIOUnpack**

7.172.1 Detailed Description

Concrete implementation of LCWriter using **SIO** (p. 30).

Use LCFactory to instantiate.

See Also

LCFactory

Author

gaede

Version

Mar 6, 2003

7.172.2 Member Function Documentation

7.172.2.1 void SIO::SIOWriter::close ( ) throw IO::IOException, std::exception) [virtual]

Closes the output file/stream etc.

Exceptions

IOException	

Implements **IO::LCWriter** (p. 167).

References SIO::LCIORandomAccessMgr::clear(), and SIO::LCIORandomAccessMgr::writeRandomAccess-Records().

7.172.2.2 void SIO::SIOWriter::flush ( ) throw IO::IOException, std::exception) [virtual]

Flushes the output file/stream etc.

Exceptions

IOException	

Implements **IO::LCWriter** (p. 167).

7.172.2.3 void SIO::SIOWriter::open (const std::string & *filename*) throw IO::IOException, std::exception) [virtual]

Opens a file for writing.

If file with given name exists, an exception is thrown. Use append or new mode instead.

Exceptions

IOException	

Implements **IO::LCWriter** (p. 167).

7.172.2.4 void SIO::SIOWriter::open (const std::string & *filename*, int *writeMode*) throw IO::IOException, std::exception) [virtual]

Opens a file for writing.

Possible write modes are: LCIO::WRITE_NEW (existing files are replaced) and LCIO::WRITE_APPEND.

Exceptions

IOException	

Implements **IO::LCWriter** (p. 168).

References SIO::LCSIO::getValidSIOName(), and SIO::LCSIO::seekStream().

7.172.2.5 void SIO::SIOWriter::setCompressionLevel (int *level*) [virtual]

Set the compression level - needs to be called before **open()** (p. 256) otherwise call will have no effect.

If not called the Writer will use default compression.

Valid compression levels are:

- level $<$ 0 : default compression
- level == 0 : no compression
- level $>$ 0 : 1 (fastest) - 9 (best compression)

Experimental code - don't use for production.

Parameters

level	compression level

Implements **IO::LCWriter** (p. 168).

7.172.2.6 void SIO::SIOWriter::setUpHandlers (const EVENT::LCEvent ∗ *evt*) [protected]

Sets up the handlers for writing the current event.

Creates Handlers needed for writing the event on this stream.

Needs to be called for every event.

References EVENT::LCEvent::getCollection(), EVENT::LCEvent::getCollectionNames(), EVENT::LCCollection-::getTypeName(), and EVENT::LCCollection::isTransient().

7.172.2.7 void SIO::SIOWriter::writeEvent (const EVENT::LCEvent ∗ *evt*) throw IO::IOException, std::exception) [virtual]

Writes the given event to file.

Exceptions

IOException	

Implements **IO::LCWriter** (p. 168).

7.172.2.8 void SIO::SIOWriter::writeRunHeader (const EVENT::LCRunHeader ∗ *hdr*) throw IO::IOException, std::exception) [virtual]

Writes the given run header to file.

Exceptions

IOException	

Implements **IO::LCWriter** (p. 168).

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

- SIOWriter.h
- SIOWriter.cc

7.173 SomeClass Struct Reference

Classes

- struct **Index**

Public Attributes

- double **someDouble**
- int **someInt**
- bool **someBool**

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

- test_RTextension.cc

7.174 TEST Class Reference

Public Member Functions

- **TEST** (const std::string &tname, std::ostream &stream=std::cout)
- void **LOG** (const std::string &msg)
- template<class V1 , class V2 >
  void **operator()** (const V1 &v1, const V2 &v2, const std::string name)
- void **FAILED** (const std::string &msg)

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

- tutil.h

7.175 lcrtrel_helper::ToRelation< U > Struct Template Reference

Helper class for relations.

`#include <LCRTRelations.h>`

7.175.1 Detailed Description

template<class U>struct lcrtrel_helper::ToRelation< U >

Helper class for relations.

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

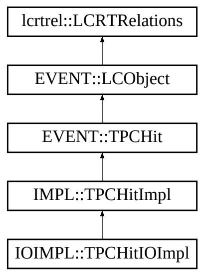
- LCRTRelations.h

7.176 EVENT::TPCHit Class Reference

Deprecated.

`#include <pre-generated/EVENT/TPCHit.h>`

Inheritance diagram for EVENT::TPCHit:



Public Types

- typedef **TPCHit lcobject_type**

  *Useful typedef for template programming with **LCIO** (p. 115).*

Public Member Functions

- virtual ∼**TPCHit** ()

  Destructor.
- virtual int **getCellID** () const =0

  Returns the detector specific cell id.
- virtual float **getTime** () const =0

  Returns the time of the hit.
- virtual float **getCharge** () const =0

  Returns the integrated charge of the hit.
- virtual int **getQuality** () const =0

  Returns a quality flag for the hit.
- virtual int **getNRawDataWords** () const =0

  Return the number of raw data (32-bit) words stored for the hit.
- virtual int **getRawDataWord** (int i) const =0

  Return the raw data (32-bit) word at i.

Additional Inherited Members

7.176.1 Detailed Description

Deprecated.

Please use **TrackerRawData** (p. 294), **TrackerData** (p. 267) and **TrackerPulse** (p. 289) instead. A future version of **LCIO** (p. 115) will not support writing of **TPCHit** (p. 258) collections anymore.

Author

Deprecated gaede

Version

Id:

TPCHit.aid,v 1.5 2006-03-24 13:25:52 gaede Exp

7.176.2 Member Function Documentation

7.176.2.1 virtual int EVENT::TPCHit::getNRawDataWords ( ) const [pure virtual]

Return the number of raw data (32-bit) words stored for the hit.

Check the flag word (bit TPCBIT_RAW) of the collection if raw data is stored at all.

Implemented in **IMPL::TPCHitImpl** (p. 261).

Referenced by UTIL::LCTOOLS::printTPCHits(), and SIO::SIOTPCHitHandler::write().

7.176.2.2 virtual int EVENT::TPCHit::getRawDataWord (int *i*) const [pure virtual]

Return the raw data (32-bit) word at i.

Check the flag word (bit TPCBIT_RAW) of the collection if raw data is stored at all.

Implemented in **IMPL::TPCHitImpl** (p. 262).

Referenced by UTIL::LCTOOLS::printTPCHits(), and SIO::SIOTPCHitHandler::write().

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

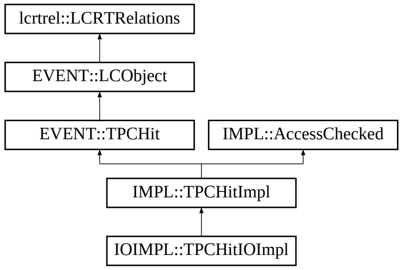
- TPCHit.h

7.177 IMPL::TPCHitImpl Class Reference

Implementation of the real data TPCHit.

`#include <IMPL/TPCHitImpl.h>`

Inheritance diagram for IMPL::TPCHitImpl:



Public Member Functions

- **TPCHitImpl** ()

  Default Constructor - initializes all data to 0's.
- virtual ∼**TPCHitImpl** ()

  Destructor.
- virtual int **id** () const

  Returns an object id for internal (debugging) use in LCIO.
- virtual int **getCellID** () const

  Returns the detector specific cell id.
- virtual float **getTime** () const

  Returns the time of the hit.
- virtual float **getCharge** () const

  Returns the integrated charge of the hit.
- virtual int **getQuality** () const

  Returns a quality flag for the hit.
- virtual int **getNRawDataWords** () const

  Return the number of raw data (32-bit) words stored for the hit.
- virtual int **getRawDataWord** (int i) const

  Return the raw data (32-bit) word at i.

- virtual void **setCellID** (int cellID)

  Set the cell id.
- virtual void **setTime** (float time)

  Set the time.
- virtual void **setCharge** (float charge)

  Set the charge.
- virtual void **setQuality** (int quality)

  Set the cell quality word.
- virtual void **setRawData** (const int ∗raw, int size)

  Set the raw data.

Protected Member Functions

- virtual void **initRawArray** (int size)

  initialize the raw data array - for faster reading

Protected Attributes

- int **_cellID**
- float **_time**
- float **_charge**
- int **_quality**
- int **_rawSize**
- int ∗ **_rawArray**

Additional Inherited Members

7.177.1 Detailed Description

Implementation of the real data TPCHit.

Author

gaede

Version

Sep 11, 2003

7.177.2 Member Function Documentation

7.177.2.1 int IMPL::TPCHitImpl::getNRawDataWords ( ) const [virtual]

Return the number of raw data (32-bit) words stored for the hit.

Check the flag word (bit TPCBIT_RAW) of the collection if raw data is stored at all.

Implements **EVENT::TPCHit** (p. 259).

7.177.2.2 int IMPL::TPCHitImpl::getRawDataWord (int *i*) const [virtual]

Return the raw data (32-bit) word at i.

Check the flag word (bit TPCBIT_RAW) of the collection if raw data is stored at all.

Implements **EVENT::TPCHit** (p. 260).

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

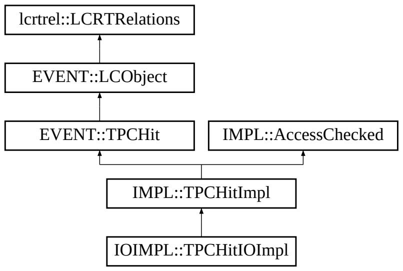
- TPCHitImpl.h
- TPCHitImpl.cc

7.178 IOIMPL::TPCHitIOImpl Class Reference

Adding stuff needed for io (friend declarations, etc.)

`#include <IOIMPL/TPCHitIOImpl.h>`

Inheritance diagram for IOIMPL::TPCHitIOImpl:



Friends

- class **SIO::SIOTPCHitHandler**

Additional Inherited Members

7.178.1 Detailed Description

Adding stuff needed for io (friend declarations, etc.)

Author

gaede

Version

Sep 11, 2003

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

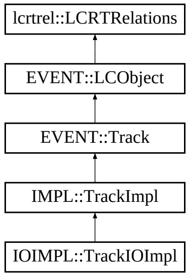
- TPCHitIOImpl.h

7.179 EVENT::Track Class Reference

The **LCIO** (p. 115) track class.

`#include <pre-generated/EVENT/Track.h>`

Inheritance diagram for EVENT::Track:



Public Types

- typedef **Track lcobject_type**

    *Useful typedef for template programming with **LCIO** (p. 115).*

Public Member Functions

- virtual ∼**Track** ()

    Destructor.
- virtual int **getType** () const =0

    Flagword that defines the type of track.
- virtual float **getD0** () const =0

    Impact paramter of the track in (r-phi).
- virtual float **getPhi** () const =0

    Phi of the track at the reference point.
- virtual float **getOmega** () const =0

    Omega is the signed curvature of the track in [1/mm].
- virtual float **getZ0** () const =0

    Impact paramter of the track in (r-z).
- virtual float **getTanLambda** () const =0

    Lambda is the dip angle of the track in r-z at the reference point.
- virtual const **FloatVec** & **getCovMatrix** () const =0

    Covariance matrix of the track parameters.
- virtual const float ∗ **getReferencePoint** () const =0

    Reference point of the track parameters.
- virtual float **getChi2** () const =0

    True if the reference point is the point of closest approach.
- virtual int **getNdf** () const =0

    Number of degrees of freedom of the track fit.
- virtual float **getdEdx** () const =0

    dEdx of the track.

- virtual float **getdEdxError** () const =0

    Error of dEdx.
- virtual float **getRadiusOfInnermostHit** () const =0

    The radius of the innermost hit that has been used in the track fit.
- virtual const **IntVec** & **getSubdetectorHitNumbers** () const =0

    A vector that holds the number of hits in particular subdetectors.
- virtual const **TrackVec** & **getTracks** () const =0

    The tracks that have been combined to this track.
- virtual const **TrackStateVec** & **getTrackStates** () const =0

    Returns track states associtated to this track.
- virtual const **TrackState** ∗ **getClosestTrackState** (float x, float y, float z) const =0

    Returns track state closest to the given point.
- virtual const **TrackState** ∗ **getTrackState** (int location) const =0

    Returns track state for the given location - or NULL if not found.
- virtual const **TrackerHitVec** & **getTrackerHits** () const =0

    Optionaly (check/set flag(LCIO::TRBIT_HITS)==1) return the hits that have been used to create this track.

Additional Inherited Members

7.179.1 Detailed Description

The **LCIO** (p. 115) track class.

The helix parameterization follows that of the L3 experiment at LEP. A detailed description of the track parameters is given in the LC-Note LC-DET-2006-004 "Track Parameters in LCIO" `(pdf)`.

Author

gaede

Version

Id:

Track.aid,v 1.15 2007-09-18 09:47:09 gaede Exp

7.179.2 Member Function Documentation

7.179.2.1 virtual float EVENT::Track::getChi2 ( ) const `[pure virtual]`

True if the reference point is the point of closest approach.

Chi^2 of the track fit.

Implemented in **IMPL::TrackImpl** (p. 301).

Referenced by SIO::SIOTrackHandler::write().

7.179.2.2 virtual const TrackState∗ EVENT::Track::getClosestTrackState (float *x*, float *y*, float *z*) const `[pure virtual]`

Returns track state closest to the given point.

See Also

TrackState (p. 304).

Implemented in **IMPL::TrackImpl** (p. 301).

7.179.2.3 virtual const FloatVec& EVENT::Track::getCovMatrix ( ) const [pure virtual]

Covariance matrix of the track parameters.

Stored as lower triangle matrix where the order of parameters is: d0, phi, omega, z0, tan(lambda). So we have cov(d0,d0), cov(phi, d0), cov(phi, phi), ... Information is stored in the first **TrackState** (p. 304) of this **Track** (p. 263),

See Also

TrackState (p. 304).

Implemented in **IMPL::TrackImpl** (p. 301).

7.179.2.4 virtual float EVENT::Track::getD0 ( ) const [pure virtual]

Impact paramter of the track in (r-phi).

Information is stored in the first **TrackState** (p. 304) of this **Track** (p. 263),

See Also

TrackState (p. 304).

Implemented in **IMPL::TrackImpl** (p. 301).

7.179.2.5 virtual float EVENT::Track::getOmega ( ) const [pure virtual]

Omega is the signed curvature of the track in [1/mm].

The sign is that of the particle's charge. Information is stored in the first **TrackState** (p. 304) of this **Track** (p. 263),

See Also

TrackState (p. 304).

Implemented in **IMPL::TrackImpl** (p. 301).

7.179.2.6 virtual float EVENT::Track::getPhi ( ) const [pure virtual]

Phi of the track at the reference point.

See Also

getReferencePoint (p. 266) Information is stored in the first **TrackState** (p. 304) of this **Track** (p. 263), **TrackState** (p. 304).

Implemented in **IMPL::TrackImpl** (p. 302).

7.179.2.7 virtual float EVENT::Track::getRadiusOfInnermostHit ( ) const [pure virtual]

The radius of the innermost hit that has been used in the track fit.

Helps to detect V0 tracks with small impact paramters or heavy mesons. To be used as convenient information or if hits are not stored in the data set, e.g. DST or FastMC.

Implemented in **IMPL::TrackImpl** (p. 302).

Referenced by SIO::SIOTrackHandler::write().

7.179.2.8 virtual const float* EVENT::Track::getReferencePoint ( ) const [pure virtual]

Reference point of the track parameters.

The default for the reference point is the point of closest approach. Information is stored in the first **TrackState** (p. 304) of this **Track** (p. 263),

See Also

TrackState (p. 304).

Implemented in **IMPL::TrackImpl** (p. 302).

7.179.2.9 virtual const IntVec& EVENT::Track::getSubdetectorHitNumbers ( ) const [pure virtual]

A vector that holds the number of hits in particular subdetectors.

The mapping of indices to subdetectors is implementation dependent. To be used as convenient information or if hits are not stored in the data set, e.g. DST or FastMC. Check/set collection variable TrackSubdetectorNames for decoding the indices.

Implemented in **IMPL::TrackImpl** (p. 302).

Referenced by SIO::SIOTrackHandler::write().

7.179.2.10 virtual float EVENT::Track::getTanLambda ( ) const [pure virtual]

Lambda is the dip angle of the track in r-z at the reference point.

See Also

getReferencePoint (p. 266) Information is stored in the first **TrackState** (p. 304) of this **Track** (p. 263), **TrackState** (p. 304).

Implemented in **IMPL::TrackImpl** (p. 302).

7.179.2.11 virtual const TrackState* EVENT::Track::getTrackState (int *location*) const [pure virtual]

Returns track state for the given location - or NULL if not found.

See Also

TrackState (p. 304). location can be: TrackState::AtIP, TrackState::AtFirstHit, TrackState::AtLastHit, TrackState::AtCalorimeter, TrackState::AtVertex, TrackState::AtOther

Implemented in **IMPL::TrackImpl** (p. 303).

7.179.2.12 virtual const TrackStateVec& EVENT::Track::getTrackStates ( ) const [pure virtual]

Returns track states associtated to this track.

See Also

TrackState (p. 304).

Implemented in **IMPL::TrackImpl** (p. 303).

Referenced by SIO::SIOTrackHandler::write().

7.179.2.13 virtual int EVENT::Track::getType ( ) const [pure virtual]

Flagword that defines the type of track.

Bits 0-15 can be used to denote the subdetectors that have contributed hits used in the track fit. For the definition of the bits check/Set the collection variables TrackTypeBitNames and TrackTypeBitIndices. Bits 16-31 are used internally.

Implemented in **IMPL::TrackImpl** (p. 303).

Referenced by SIO::SIOTrackHandler::write().

7.179.2.14 virtual float EVENT::Track::getZ0 ( ) const [pure virtual]

Impact paramter of the track in (r-z).

Information is stored in the first **TrackState** (p. 304) of this **Track** (p. 263),

See Also

TrackState (p. 304).

Implemented in **IMPL::TrackImpl** (p. 303).

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

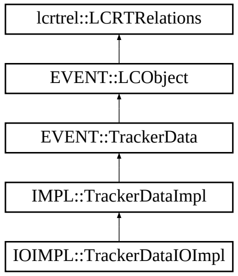
- Track.h

7.180 EVENT::TrackerData Class Reference

TrackerData (p. 267) contains the corrected (calibrated) raw tracker data.

`#include <pre-generated/EVENT/TrackerData.h>`

Inheritance diagram for EVENT::TrackerData:



Public Types

- typedef **TrackerData lcobject_type**

  *Useful typedef for template programming with **LCIO** (p. 115).*

Public Member Functions

- virtual ∼**TrackerData** ()

  Destructor.
- virtual int **getCellID0** () const =0

  Returns the first detector specific (geometrical) cell id.
- virtual int **getCellID1** () const =0

  Returns the second detector specific (geometrical) cell id.
- virtual float **getTime** () const =0

  Returns a time measurement associated with the adc values, e.g.
- virtual const **FloatVec** & **getChargeValues** () const =0

  The corrected (calibrated) FADC spectrum.

Additional Inherited Members

7.180.1 Detailed Description

TrackerData (p. 267) contains the corrected (calibrated) raw tracker data.

See Also

TrackerRawData (p. 294)
TrackerPulse (p. 289)

Author

gaede

Version

Id:

TrackerData.aid,v 1.2 2006-03-24 13:25:53 gaede Exp

7.180.2 Member Function Documentation

7.180.2.1 virtual int EVENT::TrackerData::getCellID1 ( ) const [pure virtual]

Returns the second detector specific (geometrical) cell id.

Optional, check/set flag(LCIO::TRAWBIT_ID1)==1.

Implemented in **IMPL::TrackerDataImpl** (p. 270).

Referenced by SIO::SIOTrackerDataHandler::write().

7.180.2.2 virtual float EVENT::TrackerData::getTime ( ) const [pure virtual]

Returns a time measurement associated with the adc values, e.g.

the t0 of the spectrum for the TPC. Subdetector dependent.

Implemented in **IMPL::TrackerDataImpl** (p. 269).

Referenced by SIO::SIOTrackerDataHandler::write().

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

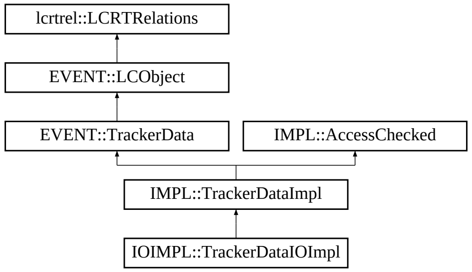
- TrackerData.h

7.181 IMPL::TrackerDataImpl Class Reference

Default implementation of TrackerData.

`#include <IMPL/TrackerDataImpl.h>`

Inheritance diagram for IMPL::TrackerDataImpl:



Public Member Functions

- **TrackerDataImpl** ()
  Default Constructor - initializes all data to 0's.
- virtual ∼**TrackerDataImpl** ()
  Destructor.
- virtual int **id** () const
  Returns an object id for internal (debugging) use in LCIO.
- virtual int **getCellID0** () const
  Returns the first detector specific (geometrical) cell id.
- virtual int **getCellID1** () const
  Returns the second detector specific (geometrical) cell id.
- virtual float **getTime** () const
  Returns the time.
- virtual const **EVENT::FloatVec** & **getChargeValues** () const
  The calibrated ADC values.
- void **setCellID0** (int cellID0)
- void **setCellID1** (int cellID1)
- void **setTime** (float time)
- void **setChargeValues** (const **EVENT::FloatVec** &charge)

  Set the charge vector by copying the values.
- **EVENT::FloatVec** & **chargeValues** ()
  Allows direct access to the charge vector.

Protected Attributes

- int **_cellID0**
- int **_cellID1**
- float **_time**
- **EVENT::FloatVec _charge**

Additional Inherited Members

7.181.1 Detailed Description

Default implementation of TrackerData.

Author

gaede

Version

Id:

TrackerDataImpl.h (p. **??**),v 1.2 2006-08-04 16:52:46 gaede Exp

7.181.2 Member Function Documentation

7.181.2.1 FloatVec & IMPL::TrackerDataImpl::chargeValues ( )

Allows direct access to the charge vector.

Can be used e.g. to set the values: raw->**chargeValues()** (p. 270).clear() ; raw->**chargeValues()** (p. 270).resize(n) ; raw->**chargeValues()** (p. 270)[i] = 42. ;

7.181.2.2 virtual int IMPL::TrackerDataImpl::getCellID1 ( ) const [inline], [virtual]

Returns the second detector specific (geometrical) cell id.

Optional, check/set flag(LCIO::TRAWBIT_ID1)==1.

Implements **EVENT::TrackerData** (p. 268).

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

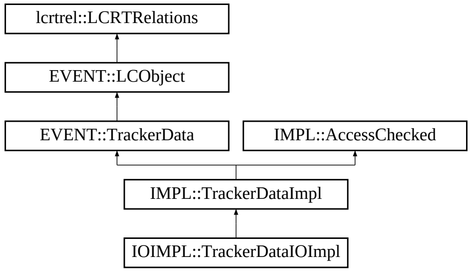
- TrackerDataImpl.h
- TrackerDataImpl.cc

7.182 IOIMPL::TrackerDataIOImpl Class Reference

Adding stuff needed for io (friend declarations, etc.)

`#include <IOIMPL/TrackerDataIOImpl.h>`

Inheritance diagram for IOIMPL::TrackerDataIOImpl:



Friends

- class **SIO::SIOTrackerDataHandler**

Additional Inherited Members

7.182.1 Detailed Description

Adding stuff needed for io (friend declarations, etc.)

Author

gaede

Version

Sep 11, 2003

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

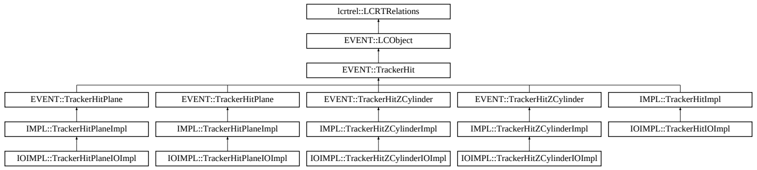
- TrackerDataIOImpl.h

7.183 EVENT::TrackerHit Class Reference

A generic tracker hit to be used by pattern recognition.

`#include <pre-generated/EVENT/TrackerHit.h>`

Inheritance diagram for EVENT::TrackerHit:



Public Types

- typedef **TrackerHit lcobject_type**

  *Useful typedef for template programming with **LCIO** (p. 115).*

Public Member Functions

- virtual ∼**TrackerHit** ()

  Destructor.
- virtual int **getCellID0** () const =0

  *Same name as in **CalorimeterHit** (p. 49), even though there are no 'cells' in this case.*
- virtual int **getCellID1** () const =0

  *Same name as in **CalorimeterHit** (p. 49), even though there are no 'cells' in this case Optional, check/set flag(LCIO-::RTHBIT_ID1)==1.*
- virtual const double ∗ **getPosition** () const =0

  The hit position in [mm].
- virtual const **FloatVec** & **getCovMatrix** () const =0

  Covariance of the position (x,y,z), stored as lower triangle matrix.
- virtual float **getdEdx** () const =0

  The dE/dx of the hit in [GeV].
- virtual float **getTime** () const =0

  The time of the hit in [ns].
- virtual int **getType** () const =0

  Type of raw data hit, either one of
  LCIO::TPCHIT
  LCIO::SIMTRACKERHIT
- virtual int **getQuality** () const =0

  The quality bit flag of the hit.
- virtual float **getEDep** () const =0

  The energy deposited on the hit [GeV].
- virtual float **getEDepError** () const =0

  The error measured on EDep [GeV].
- virtual const **LCObjectVec** & **getRawHits** () const =0

  The raw data hits.

Additional Inherited Members

7.183.1 Detailed Description

A generic tracker hit to be used by pattern recognition.

Author

gaede

Version

Id:

TrackerHit.aid,v 1.16 2010-06-17 12:28:59 engels Exp

7.183.2 Member Function Documentation

7.183.2.1 virtual const FloatVec& EVENT::TrackerHit::getCovMatrix ( ) const [pure virtual]

Covariance of the position (x,y,z), stored as lower triangle matrix.

i.e. cov(x,x) , cov(y,x) , cov(y,y) , cov(z,x) , cov(z,y) , cov(z,z).

Implemented in **IMPL::TrackerHitPlaneImpl** (p. 280), **IMPL::TrackerHitZCylinderImpl** (p. 286), and **IMPL::-TrackerHitImpl** (p. 274).

Referenced by SIO::SIOTrackerHitHandler::write().

7.183.2.2 virtual float EVENT::TrackerHit::getdEdx ( ) const [pure virtual]

The dE/dx of the hit in [GeV].

See Also

Deprecated **getEDep()** (p. 272)

Implemented in **IMPL::TrackerHitPlaneImpl** (p. 282), **IMPL::TrackerHitZCylinderImpl** (p. 287), and **IMPL::-TrackerHitImpl** (p. 276).

7.183.2.3 virtual const LCObjectVec& EVENT::TrackerHit::getRawHits ( ) const [pure virtual]

The raw data hits.

Check **getType()** (p. 273) to get actual data type.

Implemented in **IMPL::TrackerHitPlaneImpl** (p. 282), **IMPL::TrackerHitZCylinderImpl** (p. 288), and **IMPL::-TrackerHitImpl** (p. 276).

Referenced by SIO::SIOTrackerHitHandler::write(), SIO::SIOTrackerHitZCylinderHandler::write(), and SIO::SIO-TrackerHitPlaneHandler::write().

7.183.2.4 virtual int EVENT::TrackerHit::getType ( ) const [pure virtual]

Type of raw data hit, either one of

LCIO::TPCHIT

LCIO::SIMTRACKERHIT

Type of hit. Mapping of integer types to type names through collection parameters "TrackerHitTypeNames" and "TrackerHitTypeValues".

Implemented in **IMPL::TrackerHitPlaneImpl** (p. 282), **IMPL::TrackerHitZCylinderImpl** (p. 288), and **IMPL::-TrackerHitImpl** (p. 276).

Referenced by SIO::SIOTrackerHitHandler::write(), SIO::SIOTrackerHitZCylinderHandler::write(), and SIO::SIO-TrackerHitPlaneHandler::write().

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

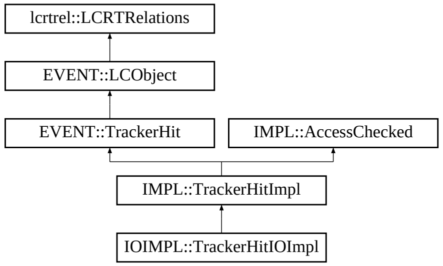
- TrackerHit.h

7.184 IMPL::TrackerHitImpl Class Reference

Implementation of the generic tracker hit.

`#include <IMPL/TrackerHitImpl.h>`

Inheritance diagram for IMPL::TrackerHitImpl:



Public Member Functions

- virtual ∼**TrackerHitImpl** ()
  Destructor.
- virtual int **id** () const
  Returns an object id for internal (debugging) use in LCIO.
- virtual int **getCellID0** () const
  Same name as in CalorimeterHit, even though there are no 'cells' in this case.
- virtual int **getCellID1** () const
  Same name as in CalorimeterHit, even though there are no 'cells' in this case 0 if information is not stored - check the flag word (bit RTHBIT_ID1) of the collection.
- virtual const double ∗ **getPosition** () const
  The hit position in [mm].
- virtual const **EVENT::FloatVec** & **getCovMatrix** () const
  Covariance of the position (x,y,z)
- virtual float **getdEdx** () const
  The dE/dx of the hit in [GeV/mm].
- virtual float **getEDep** () const
  The deposited energy of the hit [GeV].
- virtual float **getEDepError** () const
  The error measured on EDep [GeV].
- virtual float **getTime** () const
  The time of the hit in [ns].
- virtual int **getType** () const
  Type of raw data hit, either one of
  LCIO::TPCHIT
  LCIO::SIMTRACKERHIT
- virtual int **getQuality** () const
  The quality bit flag of the hit.
- virtual const **EVENT::LCObjectVec** & **getRawHits** () const
  The raw data hits.
- virtual **EVENT::LCObjectVec** & **rawHits** ()
  Use to manipulate the raw hits.
- void **setCellID0** (int id0)
  Sets the first cell id;.
- void **setCellID1** (int id1)

Sets the second cell id; Only store if the flag word (bit RTHBIT_ID1) of the collection is set.

- void **setType** (int type)
- void **setPosition** (const double pos[3])
- void **setCovMatrix** (const **EVENT::FloatVec** &cov)
- void **setCovMatrix** (const float cov[TRKHITNCOVMATRIX])
- void **setdEdx** (float dedx)
- void **setEDep** (float e)
- void **setEDepError** (float e)
- void **setTime** (float t)
- void **setQuality** (int quality)
- void **setQualityBit** (int bit, bool val=true)

Protected Attributes

- int **_cellID0**
- int **_cellID1**
- int **_type**
- double **_pos** [3]
- **EVENT::FloatVec _cov**
- float **_EDep**
- float **_EDepError**
- float **_time**
- int **_quality**
- **EVENT::LCObjectVec _rawHits**

Additional Inherited Members

7.184.1 Detailed Description

Implementation of the generic tracker hit.

Author

gaede

Version

Id:

TrackerHitImpl.h (p. **??**),v 1.15 2010-06-17 12:29:00 engels Exp

7.184.2 Member Function Documentation

7.184.2.1 int IMPL::TrackerHitImpl::getCellID1 ( ) const [virtual]

Same name as in CalorimeterHit, even though there are no 'cells' in this case 0 if information is not stored - check the flag word (bit RTHBIT_ID1) of the collection.

Default is to store only cellid0.

Implements **EVENT::TrackerHit** (p. 272).

7.184.2.2 float IMPL::TrackerHitImpl::getdEdx ( ) const [virtual]

The dE/dx of the hit in [GeV/mm].

See Also

Deprecated **getEDep()** (p. 274)

Implements **EVENT::TrackerHit** (p. 273).

References UTIL::LCWarning::printWarning().

7.184.2.3 const EVENT::LCObjectVec & IMPL::TrackerHitImpl::getRawHits ( ) const [virtual]

The raw data hits.

Check **getType()** (p. 276) to get actual data type.

Implements **EVENT::TrackerHit** (p. 273).

7.184.2.4 float IMPL::TrackerHitImpl::getTime ( ) const [virtual]

The time of the hit in [ns].

Is this needed ?

Implements **EVENT::TrackerHit** (p. 272).

7.184.2.5 int IMPL::TrackerHitImpl::getType ( ) const [virtual]

Type of raw data hit, either one of

LCIO::TPCHIT

LCIO::SIMTRACKERHIT

Type of hit. Mapping of integer types to type names through collection parameters "TrackerHitTypeNames" and "TrackerHitTypeValues".

Implements **EVENT::TrackerHit** (p. 273).

7.184.2.6 void IMPL::TrackerHitImpl::setCellID1 (int *id1*)

Sets the second cell id; Only store if the flag word (bit RTHBIT_ID1) of the collection is set.

Default is to store only cellid0.

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

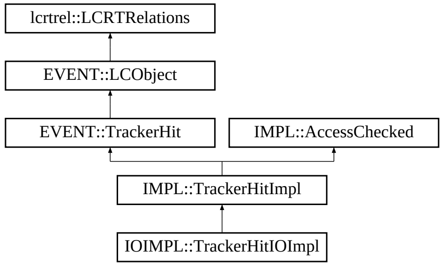
- TrackerHitImpl.h
- TrackerHitImpl.cc

7.185 IOIMPL::TrackerHitIOImpl Class Reference

Adding stuff needed for io (friend declarations, etc.)

`#include <IOIMPL/TrackerHitIOImpl.h>`

Inheritance diagram for IOIMPL::TrackerHitIOImpl:



Friends

- class **SIO::SIOTrackerHitHandler**

Additional Inherited Members

7.185.1 Detailed Description

Adding stuff needed for io (friend declarations, etc.)

Author

gaede

Version

Mar 12, 2003

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

- TrackerHitIOImpl.h

7.186 EVENT::TrackerHitPlane Class Reference

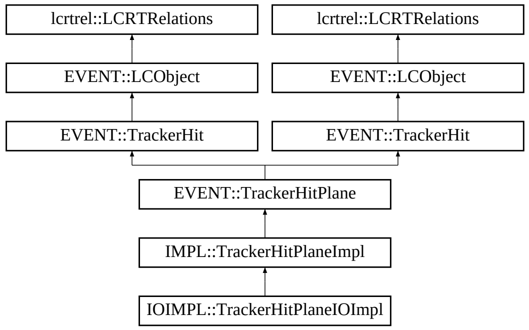
A tracker hit on a planar surface - the hit is stored as:

x,y,z, u(theta, phi), v(theta,phi), du, dv

where (x,y,z) is the position in global cartesian coordinates, u,v are the the meassurement directions (unit vectors, spanning the plane) and du,dv are the measurement errors along these directions.

`#include <pre-generated/EVENT/TrackerHitPlane.h>`

Inheritance diagram for EVENT::TrackerHitPlane:



Public Types

- typedef **TrackerHitPlane lcobject_type**

  *Useful typedef for template programming with **LCIO** (p. 115).*

Public Member Functions

- virtual ∼**TrackerHitPlane** ()

  Destructor.
- virtual int **getCellID0** () const =0

  *Same name as in **CalorimeterHit** (p. 49), even though there are no 'cells' in this case.*
- virtual int **getCellID1** () const =0

  *Same name as in **CalorimeterHit** (p. 49), even though there are no 'cells' in this case Optional, check/set flag(LCIO-::RTHPBIT_ID1)==1.*
- virtual const float ∗ **getU** () const =0

  Direction of first measurement - given as (theta, phi).
- virtual const float ∗ **getV** () const =0

  Direction of second measurement - given as (theta, phi).
- virtual float **getdU** () const =0

  Measurement error along u.
- virtual float **getdV** () const =0

  Measurement error along v.

Additional Inherited Members

7.186.1 Detailed Description

A tracker hit on a planar surface - the hit is stored as:

x,y,z, u(theta, phi), v(theta,phi), du, dv

where (x,y,z) is the position in global cartesian coordinates, u,v are the the meassurement directions (unit vectors, spanning the plane) and du,dv are the measurement errors along these directions.

The order of u and v should follow the global cartesian coordinates x, y and z, e.g. in case of a plane parallel to the z-axis, u lies in the x-y plane and v is along z.

Author

gaede

Version

$Id:$

7.186.2 Member Function Documentation

7.186.2.1 virtual const float∗ EVENT::TrackerHitPlane::getU ( ) const [pure virtual]

Direction of first measurement - given as (theta, phi).

Defines spanning vector of measurement plane.

Implemented in **IMPL::TrackerHitPlaneImpl** (p. 282).

Referenced by SIO::SIOTrackerHitPlaneHandler::write().

7.186.2.2 virtual const float∗ EVENT::TrackerHitPlane::getV ( ) const [pure virtual]

Direction of second measurement - given as (theta, phi).

Defines spanning vector of measurement plane.

Implemented in **IMPL::TrackerHitPlaneImpl** (p. 282).

Referenced by SIO::SIOTrackerHitPlaneHandler::write().

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

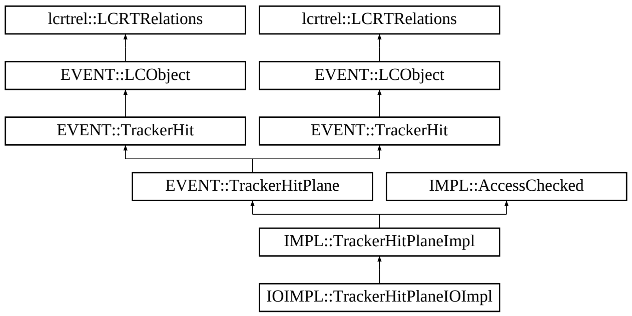
- TrackerHitPlane.h

7.187 IMPL::TrackerHitPlaneImpl Class Reference

Implementation of the planar tracker hit.

`#include <IMPL/TrackerHitPlaneImpl.h>`

Inheritance diagram for IMPL::TrackerHitPlaneImpl:



Public Member Functions

- virtual ∼**TrackerHitPlaneImpl** ()
  Destructor.
- virtual int **id** () const
  Returns an object id for internal (debugging) use in LCIO.
- virtual int **getCellID0** () const
  Same name as in CalorimeterHit, even though there are no 'cells' in this case.
- virtual int **getCellID1** () const
  Same name as in CalorimeterHit, even though there are no 'cells' in this case 0 if information is not stored - check the flag word (bit RTHPBIT_ID1) of the collection.
- virtual const double ∗ **getPosition** () const
  The hit position in [mm].
- virtual const float ∗ **getU** () const
  Direction of first measurement - given as (theta, phi).
- virtual const float ∗ **getV** () const
  Direction of second measurement - given as (theta, phi).
- virtual float **getdU** () const
  Error along u.
- virtual float **getdV** () const
  Error along v.
- virtual const **EVENT::FloatVec** & **getCovMatrix** () const
  Covariance of the position (x,y,z)
- virtual float **getdEdx** () const
  The dE/dx of the hit in [GeV/mm].
- virtual float **getEDep** () const
  The deposited energy of the hit [GeV].
- virtual float **getEDepError** () const
  The error measured on EDep [GeV].
- virtual float **getTime** () const
  The time of the hit in [ns].
- virtual int **getType** () const
  Type of hit.
- virtual int **getQuality** () const
  The quality bit flag of the hit.
- virtual const **EVENT::LCObjectVec** & **getRawHits** () const
  The raw data hits.
- virtual **EVENT::LCObjectVec** & **rawHits** ()
  Use to manipulate the raw hits.
- void **setCellID0** (int id0)
  Sets the first cell id;.
- void **setCellID1** (int id1)
  Sets the second cell id; Only store if the flag word (bit RTHPBIT_ID1) of the collection is set.
- void **setType** (int type)
- void **setPosition** (const double pos[3])
- void **setU** (const float u[2])
- void **setU** (float theta, float phi)
- void **setV** (const float v[2])
- void **setV** (float theta, float phi)
- void **setdU** (float du)
- void **setdV** (float dv)
- void **setEDep** (float e)

- void **setEDepError** (float e)
- void **setTime** (float t)
- void **setQuality** (int quality)
- void **setQualityBit** (int bit, bool val=true)

Protected Attributes

- int **_cellID0**
- int **_cellID1**
- int **_type**
- double **_pos** [3]
- float **_u** [2]
- float **_v** [2]
- float **_du**
- float **_dv**
- float **_EDep**
- float **_EDepError**
- float **_time**
- int **_quality**
- **EVENT::FloatVec _cov**
- **EVENT::LCObjectVec _rawHits**

Additional Inherited Members

7.187.1 Detailed Description

Implementation of the planar tracker hit.

Author

gaede

Version

Id:

7.187.2 Member Function Documentation

7.187.2.1 virtual int IMPL::TrackerHitPlaneImpl::getCellID1 ( ) const [inline], [virtual]

Same name as in CalorimeterHit, even though there are no 'cells' in this case 0 if information is not stored - check the flag word (bit RTHPBIT_ID1) of the collection.

Default is to store only cellid0.

Implements **EVENT::TrackerHitPlane** (p. 278).

7.187.2.2 float IMPL::TrackerHitPlaneImpl::getdEdx ( ) const [virtual]

The dE/dx of the hit in [GeV/mm].

See Also

Deprecated **getEDep()** (p. 280)

Implements **EVENT::TrackerHit** (p. 273).

References UTIL::LCWarning::printWarning().

7.187.2.3 virtual const EVENT::LCObjectVec& IMPL::TrackerHitPlaneImpl::getRawHits ( ) const [inline], [virtual]

The raw data hits.

Check **getType()** (p. 282) to get actual data type.

Implements **EVENT::TrackerHit** (p. 273).

7.187.2.4 virtual float IMPL::TrackerHitPlaneImpl::getTime ( ) const [inline], [virtual]

The time of the hit in [ns].

Is this needed ?

Implements **EVENT::TrackerHit** (p. 272).

7.187.2.5 virtual int IMPL::TrackerHitPlaneImpl::getType ( ) const [inline], [virtual]

Type of hit.

Mapping of integer types to type names through collection parameters "TrackerHitTypeNames" and "TrackerHitTypeValues".

Implements **EVENT::TrackerHit** (p. 273).

7.187.2.6 virtual const float* IMPL::TrackerHitPlaneImpl::getU ( ) const [inline], [virtual]

Direction of first measurement - given as (theta, phi).

Defines spanning vector of measurement plane.

Implements **EVENT::TrackerHitPlane** (p. 279).

7.187.2.7 virtual const float* IMPL::TrackerHitPlaneImpl::getV ( ) const [inline], [virtual]

Direction of second measurement - given as (theta, phi).

Defines spanning vector of measurement plane.

Implements **EVENT::TrackerHitPlane** (p. 279).

7.187.2.8 void IMPL::TrackerHitPlaneImpl::setCellID1 (int *id1*)

Sets the second cell id; Only store if the flag word (bit RTHPBIT_ID1) of the collection is set.

Default is to store only cellid0.

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

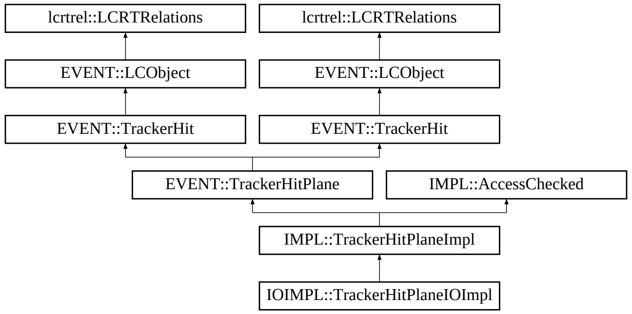
- TrackerHitPlaneImpl.h
- TrackerHitPlaneImpl.cc

7.188 IOIMPL::TrackerHitPlaneIOImpl Class Reference

Adding stuff needed for io (friend declarations, etc.)

`#include <IOIMPL/TrackerHitPlaneIOImpl.h>`

Inheritance diagram for IOIMPL::TrackerHitPlaneIOImpl:



Friends

- class **SIO::SIOTrackerHitPlaneHandler**

Additional Inherited Members

7.188.1 Detailed Description

Adding stuff needed for io (friend declarations, etc.)

Author

engels

Version

06-2011

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

- TrackerHitPlaneIOImpl.h

7.189 EVENT::TrackerHitZCylinder Class Reference

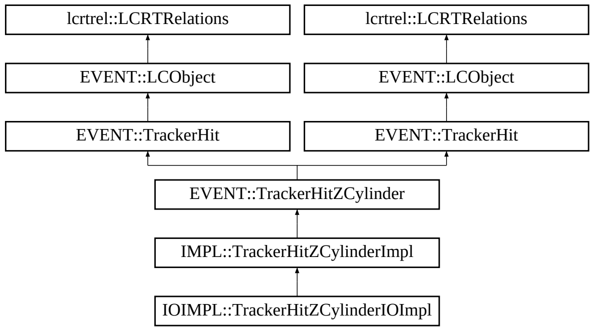
A tracker hit on a cylindrical surface that runs parallel to the z-axis - the hit is stored as:

x,y,z, center(x,y), dRPhi, dz

where (x,y,z) is the position in global cartesian coordinates, center(x,y) is the center of the cylinder and Rphi and z are the measurement errors.

`#include <pre-generated/EVENT/TrackerHitZCylinder.h>`

Inheritance diagram for EVENT::TrackerHitZCylinder:



Public Types

- typedef **TrackerHitZCylinder lcobject_type**

  Useful typedef for template programming with ***LCIO*** *(p. 115).*

Public Member Functions

- virtual ∼**TrackerHitZCylinder** ()

  Destructor.
- virtual int **getCellID0** () const =0

  Same name as in ***CalorimeterHit*** *(p. 49), even though there are no 'cells' in this case.*
- virtual int **getCellID1** () const =0

  Same name as in ***CalorimeterHit*** *(p. 49), even though there are no 'cells' in this case Optional, check/set flag(LCIO::RTHZBIT_ID1)==1.*
- virtual const float ∗ **getCenter** () const =0

  Radius of cylinder.
- virtual float **getdRPhi** () const =0

  Measurement error along RPhi.
- virtual float **getdZ** () const =0

  Measurement error along z.

Additional Inherited Members

7.189.1 Detailed Description

A tracker hit on a cylindrical surface that runs parallel to the z-axis - the hit is stored as:

x,y,z, center(x,y), dRPhi, dz

where (x,y,z) is the position in global cartesian coordinates, center(x,y) is the center of the cylinder and Rphi and z are the measurement errors.

Note the radius of the cylinder is given by R=sqrt(dx^2c+ dy^2) and dx=x-x_c, dy=y-y_c.

Author

gaede

Version

$Id:$

7.189.2 Member Function Documentation

7.189.2.1 virtual const float∗ EVENT::TrackerHitZCylinder::getCenter ( ) const [pure virtual]

Radius of cylinder.

Center of cylinder in RPhi-plane - given as (x,y).

Implemented in **IMPL::TrackerHitZCylinderImpl** (p. 287).

Referenced by SIO::SIOTrackerHitZCylinderHandler::write().

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

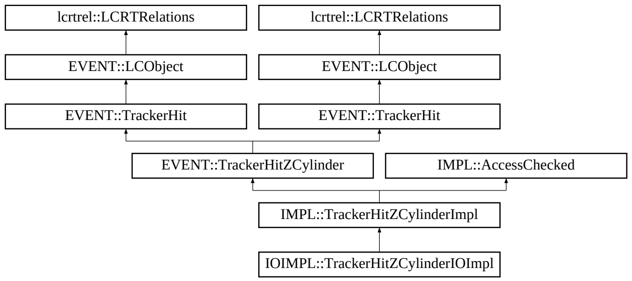
- TrackerHitZCylinder.h

7.190 IMPL::TrackerHitZCylinderImpl Class Reference

Implementation of the tracker hit on a cylindrical surface parallel to z.

`#include <IMPL/TrackerHitZCylinderImpl.h>`

Inheritance diagram for IMPL::TrackerHitZCylinderImpl:



Public Member Functions

- virtual ∼**TrackerHitZCylinderImpl** ()
  Destructor.
- virtual int **id** () const
  Returns an object id for internal (debugging) use in LCIO.
- virtual int **getCellID0** () const
  Same name as in CalorimeterHit, even though there are no 'cells' in this case.
- virtual int **getCellID1** () const
  Same name as in CalorimeterHit, even though there are no 'cells' in this case 0 if information is not stored - check the flag word (bit RTHZBIT_ID1) of the collection.

- virtual const double ∗ **getPosition** () const
  The hit position in [mm].
- virtual const float ∗ **getCenter** () const
  Radius of cylinder.
- virtual float **getdRPhi** () const
  Measurement error along RPhi.
- virtual float **getdZ** () const
  Measurement error along z.
- virtual const **EVENT::FloatVec** & **getCovMatrix** () const
  Covariance of the position (x,y,z)
- virtual float **getdEdx** () const
  The dE/dx of the hit in [GeV/mm].
- virtual float **getEDep** () const
  The deposited energy of the hit [GeV].
- virtual float **getEDepError** () const
  The error measured on EDep [GeV].
- virtual float **getTime** () const
  The time of the hit in [ns].
- virtual int **getType** () const
  Type of hit.
- virtual int **getQuality** () const
  The quality bit flag of the hit.
- virtual const **EVENT::LCObjectVec** & **getRawHits** () const
  The raw data hits.
- virtual **EVENT::LCObjectVec** & **rawHits** ()
  Use to manipulate the raw hits.
- void **setCellID0** (int id0)
  Sets the first cell id;.
- void **setCellID1** (int id1)
  Sets the second cell id; Only store if the flag word (bit RTHZBIT_ID1) of the collection is set.
- void **setType** (int type)
- void **setPosition** (const double pos[3])
- void **setCenter** (const float c[2])
- void **setCenter** (float xc, float yc)
- void **setdRPhi** (float drphi)
- void **setdZ** (float dz)
- void **setEDep** (float e)
- void **setEDepError** (float e)
- void **setTime** (float t)
- void **setQuality** (int quality)
- void **setQualityBit** (int bit, bool val=true)

Protected Attributes

- int **_cellID0**
- int **_cellID1**
- int **_type**
- double **_pos** [3]
- float **_center** [2]
- float **_drphi**
- float **_dz**
- float **_EDep**

- float **_EDepError**
- float **_time**
- int **_quality**
- **EVENT::FloatVec _cov**
- **EVENT::LCObjectVec _rawHits**

Additional Inherited Members

7.190.1 Detailed Description

Implementation of the tracker hit on a cylindrical surface parallel to z.

Author

gaede

Version

Id:

7.190.2 Member Function Documentation

7.190.2.1 virtual int IMPL::TrackerHitZCylinderImpl::getCellID1 ( ) const [inline], [virtual]

Same name as in CalorimeterHit, even though there are no 'cells' in this case 0 if information is not stored - check the flag word (bit RTHZBIT_ID1) of the collection.

Default is to store only cellid0.

Implements **EVENT::TrackerHitZCylinder** (p. 284).

7.190.2.2 virtual const float∗ IMPL::TrackerHitZCylinderImpl::getCenter ( ) const [inline], [virtual]

Radius of cylinder.

Center of cylinder in RPhi-plane - given as (x,y).

Implements **EVENT::TrackerHitZCylinder** (p. 285).

7.190.2.3 float IMPL::TrackerHitZCylinderImpl::getdEdx ( ) const [virtual]

The dE/dx of the hit in [GeV/mm].

See Also

Deprecated **getEDep()** (p. 286)

Implements **EVENT::TrackerHit** (p. 273).

References UTIL::LCWarning::printWarning().

7.190.2.4 virtual const EVENT::LCObjectVec& IMPL::TrackerHitZCylinderImpl::getRawHits ( ) const [inline], [virtual]

The raw data hits.

Check **getType()** (p. 288) to get actual data type.

Implements **EVENT::TrackerHit** (p. 273).

7.190.2.5 virtual float IMPL::TrackerHitZCylinderImpl::getTime ( ) const [inline], [virtual]

The time of the hit in [ns].

Is this needed ?

Implements **EVENT::TrackerHit** (p. 272).

7.190.2.6 virtual int IMPL::TrackerHitZCylinderImpl::getType ( ) const [inline], [virtual]

Type of hit.

Mapping of integer types to type names through collection parameters "TrackerHitTypeNames" and "TrackerHitTypeValues".

Implements **EVENT::TrackerHit** (p. 273).

7.190.2.7 void IMPL::TrackerHitZCylinderImpl::setCellID1 (int *id1*)

Sets the second cell id; Only store if the flag word (bit RTHZBIT_ID1) of the collection is set.

Default is to store only cellid0.

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

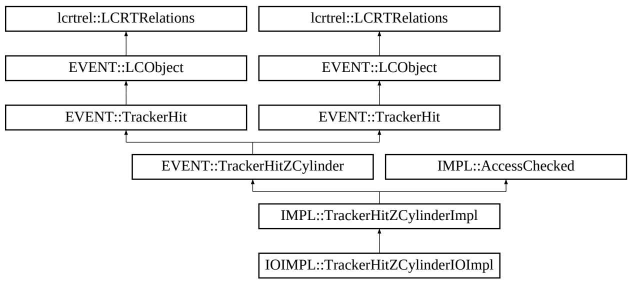
- TrackerHitZCylinderImpl.h
- TrackerHitZCylinderImpl.cc

7.191 IOIMPL::TrackerHitZCylinderIOImpl Class Reference

Adding stuff needed for io (friend declarations, etc.)

`#include <IOIMPL/TrackerHitZCylinderIOImpl.h>`

Inheritance diagram for IOIMPL::TrackerHitZCylinderIOImpl:



Friends

- class **SIO::SIOTrackerHitZCylinderHandler**

Additional Inherited Members

7.191.1 Detailed Description

Adding stuff needed for io (friend declarations, etc.)

Author

engels

Version

2011-06

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

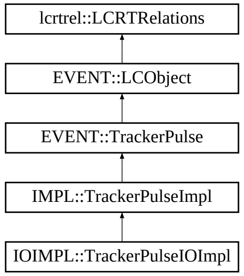
- TrackerHitZCylinderIOImpl.h

7.192 EVENT::TrackerPulse Class Reference

Tracker pulses as computed from **TrackerData** (p. 267) objects or as directly measured by a specific subdetector.

`#include <pre-generated/EVENT/TrackerPulse.h>`

Inheritance diagram for EVENT::TrackerPulse:



Public Types

- typedef **TrackerPulse lcobject_type**

  *Useful typedef for template programming with **LCIO** (p. 115).*

Public Member Functions

- virtual ~**TrackerPulse** ()

  Destructor.
- virtual int **getCellID0** () const =0

  Returns the first detector specific (geometrical) cell id.
- virtual int **getCellID1** () const =0

  Returns the second detector specific (geometrical) cell id.
- virtual float **getTime** () const =0

  The time of the pulse - arbitrary units.
- virtual float **getCharge** () const =0

  The integrated charge of the pulse - arbitrary units.
- virtual const **FloatVec** & **getCovMatrix** () const =0

  Covariance matrix of the charge (c) and time (t) measurements.
- virtual int **getQuality** () const =0

  The quality bit flag of the pulse - check/set collection parameters TrackerPulseQualityNames and TrackerPulseQuality-NamesValues.
- virtual **TrackerData** ∗ **getTrackerData** () const =0

  *Optionally the **TrackerData** (p. 267) that has been used to create the pulse can be stored with the pulse - NULL if none.*

Additional Inherited Members

7.192.1 Detailed Description

Tracker pulses as computed from **TrackerData** (p. 267) objects or as directly measured by a specific subdetector.

Typically TrackerHIts are created from TrackerPulses by some sort of clustering.

See Also

TrackerRawData (p. 294)
TrackerData (p. 267)

Author

gaede

Version

Id:

TrackerPulse.aid,v 1.5 2010-11-11 09:35:27 engels Exp

7.192.2 Member Function Documentation

7.192.2.1 virtual int EVENT::TrackerPulse::getCellID1 ( ) const [pure virtual]

Returns the second detector specific (geometrical) cell id.

Optional, check/set flag(LCIO::TRAWBIT_ID1)==1.

Implemented in **IMPL::TrackerPulseImpl** (p. 292).

Referenced by SIO::SIOTrackerPulseHandler::write().

7.192.2.2 virtual const FloatVec& EVENT::TrackerPulse::getCovMatrix ( ) const `[pure virtual]`

Covariance matrix of the charge (c) and time (t) measurements.

Stored as lower triangle matrix, i.e. cov(c,c) , cov(t,c) , cov(t,t). Optional, check/set flag(LCIO::TRAWBIT_CM)==1.

Implemented in **IMPL::TrackerPulseImpl** (p. 293).

Referenced by SIO::SIOTrackerPulseHandler::write().

7.192.2.3 virtual TrackerData∗ EVENT::TrackerPulse::getTrackerData ( ) const `[pure virtual]`

Optionally the **TrackerData** (p. 267) that has been used to create the pulse can be stored with the pulse - NULL if none.

Check the quality bits for reason why the spectrum has been stored for the pulse.

Implemented in **IMPL::TrackerPulseImpl** (p. 293).

Referenced by SIO::SIOTrackerPulseHandler::write().

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

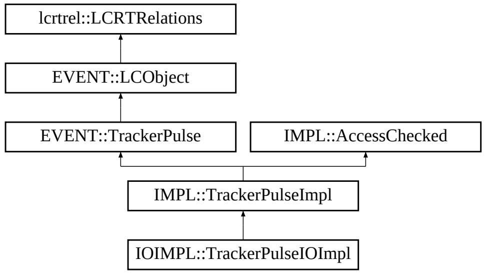
- TrackerPulse.h

7.193 IMPL::TrackerPulseImpl Class Reference

Default implementation of TrackerPulse.

`#include <IMPL/TrackerPulseImpl.h>`

Inheritance diagram for IMPL::TrackerPulseImpl:



Public Member Functions

- **TrackerPulseImpl** ()
  Default Constructor - initializes all data to 0's.
- virtual ∼**TrackerPulseImpl** ()
  Destructor.
- virtual int **id** () const
  Returns an object id for internal (debugging) use in LCIO.
- virtual int **getCellID0** () const
  Returns the first detector specific (geometrical) cell id.
- virtual int **getCellID1** () const

  Returns the second detector specific (geometrical) cell id.
- virtual float **getTime** () const
  The time of the pulse.
- virtual float **getCharge** () const
  The integrated charge of the pulse // FIXME: unit ?.
- virtual const **EVENT::FloatVec** & **getCovMatrix** () const
  Covariance matrix of the charge (c) and time (t) measurements.
- virtual int **getQuality** () const
  The quality bit flag of the pulse - use the defined constants for referring to the bits.
- virtual **EVENT::TrackerData** ∗ **getTrackerData** () const
  Optionally the TrackerData that has been used to create the pulse can be stored with the pulse - NULL if none.
- void **setCellID0** (int cellID0)
- void **setCellID1** (int cellID1)
- void **setTime** (float time)
- void **setCharge** (float charge)
- void **setCovMatrix** (const float ∗cov)
- void **setCovMatrix** (const **EVENT::FloatVec** &)
- void **setQuality** (int quality)
- void **setQualityBit** (int bit, bool val=true)
- void **setTrackerData** (**EVENT::TrackerData** ∗corrData)

Protected Attributes

- int **_cellID0**
- int **_cellID1**
- float **_time**
- float **_charge**
- int **_quality**
- **EVENT::FloatVec _cov**
- **EVENT::TrackerData** ∗ **_corrData**

Additional Inherited Members

7.193.1 Detailed Description

Default implementation of TrackerPulse.

Author

gaede

Version

Id:

TrackerPulseImpl.h (p. **??**),v 1.4 2010-05-12 14:50:00 engels Exp

7.193.2 Member Function Documentation

7.193.2.1 virtual int IMPL::TrackerPulseImpl::getCellID1 ( ) const `[inline]`, `[virtual]`

Returns the second detector specific (geometrical) cell id.

Optional, check/set flag(LCIO::TRAWBIT_ID1)==1.

Implements **EVENT::TrackerPulse** (p. 290).

7.193.2.2 virtual const **EVENT::FloatVec**& IMPL::TrackerPulseImpl::getCovMatrix ( ) const [inline], [virtual]

Covariance matrix of the charge (c) and time (t) measurements.

Stored as lower triangle matrix, i.e. cov(c,c) , cov(t,c) , cov(t,t). Optional, check/set flag(LCIO::TRAWBIT_CM)==1.

Implements **EVENT::TrackerPulse** (p. 291).

7.193.2.3 virtual **EVENT::TrackerData**∗ IMPL::TrackerPulseImpl::getTrackerData ( ) const [inline], [virtual]

Optionally the TrackerData that has been uesed to create the pulse can be stored with the pulse - NULL if none.

Check the quality bits for reason why the spectrum has been stored for the pulse.

Implements **EVENT::TrackerPulse** (p. 291).

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

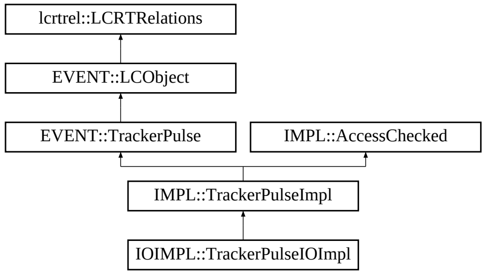
- TrackerPulseImpl.h
- TrackerPulseImpl.cc

7.194 IOIMPL::TrackerPulseIOImpl Class Reference

Adding stuff needed for io (friend declarations, etc.)

`#include <IOIMPL/TrackerPulseIOImpl.h>`

Inheritance diagram for IOIMPL::TrackerPulseIOImpl:



Friends

- class **SIO::SIOTrackerPulseHandler**

Additional Inherited Members

7.194.1 Detailed Description

Adding stuff needed for io (friend declarations, etc.)

Author

gaede

Version

Sep 11, 2003

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

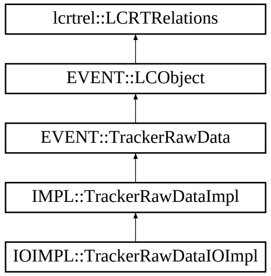
- TrackerPulseIOImpl.h

7.195 EVENT::TrackerRawData Class Reference

Generic class for raw tracker data.

`#include <pre-generated/EVENT/TrackerRawData.h>`

Inheritance diagram for EVENT::TrackerRawData:



Public Types

- typedef **TrackerRawData lcobject_type**
  Useful typedef for template programming with ***LCIO*** *(p. 115).*

Public Member Functions

- virtual ∼**TrackerRawData** ()
  Destructor.
- virtual int **getCellID0** () const =0
  Returns the first detector specific (geometrical) cell id.
- virtual int **getCellID1** () const =0
  Returns the second detector specific (geometrical) cell id.
- virtual int **getTime** () const =0
  Returns a time measurement associated with the adc values, e.g.
- virtual const **ShortVec** & **getADCValues** () const =0
  The actual FADC spectrum.

Additional Inherited Members

7.195.1 Detailed Description

Generic class for raw tracker data.

It can be used to store the full FADC spectrum as it comes out of the tracker DAQ or just one ore more single FADC readout values.

See Also

TrackerData (p. 267)
TrackerPulse (p. 289)

Author

gaede

Version

Id:

TrackerRawData.aid,v 1.2 2006-03-24 13:25:53 gaede Exp

7.195.2 Member Function Documentation

7.195.2.1 virtual int EVENT::TrackerRawData::getCellID1 ( ) const [pure virtual]

Returns the second detector specific (geometrical) cell id.

Optional, check/set flag(LCIO::TRAWBIT_ID1)==1.

Implemented in **IMPL::TrackerRawDataImpl** (p. 297).

Referenced by SIO::SIOTrackerRawDataHandler::write().

7.195.2.2 virtual int EVENT::TrackerRawData::getTime ( ) const [pure virtual]

Returns a time measurement associated with the adc values, e.g.

the t0 of the spectrum for the TPC. Subdetector dependent.

Implemented in **IMPL::TrackerRawDataImpl** (p. 296).

Referenced by SIO::SIOTrackerRawDataHandler::write().

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

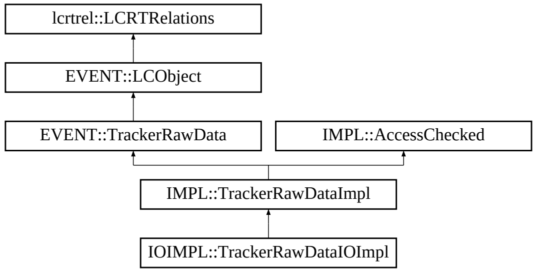
- TrackerRawData.h

7.196 IMPL::TrackerRawDataImpl Class Reference

Default implementation of TrackerRawData.

`#include <IMPL/TrackerRawDataImpl.h>`

Inheritance diagram for IMPL::TrackerRawDataImpl:



Public Member Functions

- **TrackerRawDataImpl** ()
  Default Constructor - initializes all data to 0's.
- virtual ∼**TrackerRawDataImpl** ()
  Destructor.
- virtual int **id** () const
  Returns an object id for internal (debugging) use in LCIO.
- virtual int **getCellID0** () const
  Returns the first detector specific (geometrical) cell id.
- virtual int **getCellID1** () const
  Returns the second detector specific (geometrical) cell id.
- virtual int **getTime** () const
  Returns the time.
- virtual const **EVENT::ShortVec** & **getADCValues** () const
  The measured ADC values.
- void **setCellID0** (int cellID0)
- void **setCellID1** (int cellID1)
- void **setTime** (int time)
- void **setADCValues** (const **EVENT::ShortVec** &adc)
  Set the ADC vector by copying the values.
- **EVENT::ShortVec** & **adcValues** ()
  Allows direct access to the adc vector.

Protected Attributes

- int **_cellID0**
- int **_cellID1**
- int **_channelID**
- int **_time**
- **EVENT::ShortVec _adc**

Additional Inherited Members

7.196.1 Detailed Description

Default implementation of TrackerRawData.

Author

gaede

Version

Id:

TrackerRawDataImpl.h (p. **??**),v 1.2 2006-08-04 16:52:47 gaede Exp

7.196.2 Member Function Documentation

7.196.2.1 ShortVec & IMPL::TrackerRawDataImpl::adcValues ( )

Allows direct access to the adc vector.

Can be used e.g. to set the values: raw->**adcValues()** (p. 297).clear() ; raw->**adcValues()** (p. 297).resize(n) ; raw->**adcValues()** (p. 297)[i] = 42 ;

7.196.2.2 virtual int IMPL::TrackerRawDataImpl::getCellID1 ( ) const [inline], [virtual]

Returns the second detector specific (geometrical) cell id.

Optional, check/set flag(LCIO::TRAWBIT_ID1)==1.

Implements **EVENT::TrackerRawData** (p. 295).

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

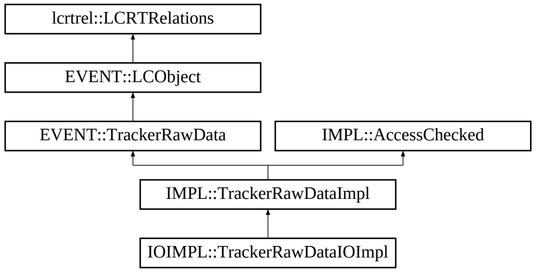
- TrackerRawDataImpl.h
- TrackerRawDataImpl.cc

7.197 IOIMPL::TrackerRawDataIOImpl Class Reference

Adding stuff needed for io (friend declarations, etc.)

`#include <IOIMPL/TrackerRawDataIOImpl.h>`

Inheritance diagram for IOIMPL::TrackerRawDataIOImpl:



Friends

- class **SIO::SIOTrackerRawDataHandler**

Additional Inherited Members

7.197.1 Detailed Description

Adding stuff needed for io (friend declarations, etc.)

Author

gaede

Version

Sep 11, 2003

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

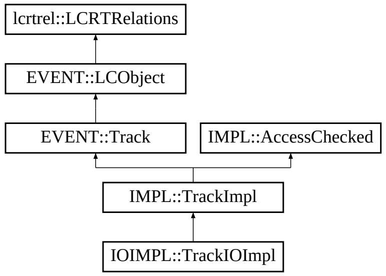
- TrackerRawDataIOImpl.h

7.198 IMPL::TrackImpl Class Reference

Implementation of the LCIO track class.

`#include <IMPL/TrackImpl.h>`

Inheritance diagram for IMPL::TrackImpl:



Public Member Functions

- **TrackImpl** ()
  Default constructor, initializes values to 0.
- **TrackImpl** (const **TrackImpl** &)
  Copy constructor - creates shallow copy, i.e.
- const **TrackImpl** & **operator=** (const **TrackImpl** &o)
- virtual ∼**TrackImpl** ()
  Destructor.
- virtual int **id** () const
  Returns an object id for internal (debugging) use in LCIO.
- virtual int **getType** () const
  Type of track, i.e.
- virtual float **getD0** () const
  Returns true if the corresponding bit in the type word is set.

- virtual float **getPhi** () const
  Phi of the track at reference point.
- virtual float **getOmega** () const
  Omega is the signed curvature of the track in [1/mm].
- virtual float **getZ0** () const
  Impact paramter of the track in (r-z).
- virtual float **getTanLambda** () const
  Lambda is the dip angle of the track in r-z at the reference point.
- virtual const **EVENT::FloatVec** & **getCovMatrix** () const
  Covariance matrix of the track parameters.
- virtual const float ∗ **getReferencePoint** () const
  Reference point of the track parameters.
- virtual float **getChi2** () const
  True if the reference point is the point of closest approach.
- virtual int **getNdf** () const
  Number of degrees of freedom of the track fit.
- virtual float **getdEdx** () const
  dEdx of the track.
- virtual float **getdEdxError** () const
  Error of dEdx.
- virtual float **getRadiusOfInnermostHit** () const
  The radius of the innermost hit that has been used in the track fit.
- virtual const **EVENT::IntVec** & **getSubdetectorHitNumbers** () const
  A vector that holds the number of hits in particular subdetectors.
- virtual const **EVENT::TrackVec** & **getTracks** () const
  The tracks (as Track objects) that have been combined to this track.
- virtual const
  EVENT::TrackStateVec & **getTrackStates** () const
  Returns track states associated with this track.
- virtual const **EVENT::TrackState** ∗ **getClosestTrackState** (float x, float y, float z) const
  Returns track state closest to the given point.
- virtual const **EVENT::TrackState** ∗ **getTrackState** (int location) const
  Returns track state for the given location - or NULL if not found.
- virtual const
  EVENT::TrackerHitVec & **getTrackerHits** () const
  Optionaly (check/set flag(LCIO::TRBIT_HITS)==1) return the hits that have been used to create this track.
- virtual void **setTypeBit** (int index, bool val=true)
- virtual void **setD0** (float d0)
- virtual void **setPhi** (float phi)
- virtual void **setOmega** (float omega)
- virtual void **setZ0** (float z0)
- virtual void **setTanLambda** (float tanLambda)
- virtual void **setCovMatrix** (const float ∗cov)
- virtual void **setCovMatrix** (const **EVENT::FloatVec** &cov)
- virtual void **setReferencePoint** (const float ∗rPnt)
- virtual void **setChi2** (float chi2)
- virtual void **setNdf** (int ndf)
- virtual void **setdEdx** (float dEdx)
- virtual void **setdEdxError** (float dEdxError)
- virtual void **addTrack** (**EVENT::Track** ∗trk)
- virtual void **addTrackState** (**EVENT::TrackState** ∗trkstate)
- virtual void **addHit** (**EVENT::TrackerHit** ∗hit)

- virtual **EVENT::TrackStateVec** & **trackStates** ()
- virtual void **setRadiusOfInnermostHit** (float r)
- virtual **EVENT::IntVec** & **subdetectorHitNumbers** ()
  Allows modification of the subdetectorHitNumbers, e.g.

Protected Member Functions

- virtual void **setType** (int type)

Protected Attributes

- std::bitset< 32 > **_type**
- float **_chi2**
- int **_ndf**
- float **_dEdx**
- float **_dEdxError**
- float **_radiusOfInnermostHit**
- **EVENT::IntVec _subdetectorHitNumbers**
- **EVENT::TrackVec _tracks**
- **EVENT::TrackerHitVec _hits**
- **EVENT::TrackStateVec _trackStates**

Additional Inherited Members

7.198.1 Detailed Description

Implementation of the LCIO track class.

The helix parameterization follows that of the L3 experiment at LEP. A detailed description of the track parameters is given in the LC-Note LC-DET-2006-004 "Track Parameters in LCIO" `(pdf)`.

See Also

Track

Author

gaede

Version

Id:

TrackImpl.h (p. **??**),v 1.15 2007-09-18 09:47:10 gaede Exp

7.198.2 Constructor & Destructor Documentation

7.198.2.1 IMPL::TrackImpl::TrackImpl (const TrackImpl & *o*)

Copy constructor - creates shallow copy, i.e.

all data members are copied but pointers to other LCObjects i.e. TrackerHits and Tracks are preserved.

7.198.3 Member Function Documentation

7.198.3.1 float IMPL::TrackImpl::getChi2 ( ) const [virtual]

True if the reference point is the point of closest approach.

Chi^2 of the track fit.

Implements **EVENT::Track** (p. 264).

7.198.3.2 const TrackState * IMPL::TrackImpl::getClosestTrackState (float *x*, float *y*, float *z*) const [virtual]

Returns track state closest to the given point.

See Also

TrackState.

Implements **EVENT::Track** (p. 264).

7.198.3.3 const FloatVec & IMPL::TrackImpl::getCovMatrix ( ) const [virtual]

Covariance matrix of the track parameters.

Stored as lower triangle matrix where the order of parameters is: d0, phi, omega, z0, tan(lambda). So we have cov(d0,d0), cov(phi, d0), cov(phi, phi), ... Information is stored in the first TrackState of this Track,

See Also

TrackState.

Implements **EVENT::Track** (p. 265).

References IMPL::TrackStateImpl::getCovMatrix().

7.198.3.4 float IMPL::TrackImpl::getD0 ( ) const [virtual]

Returns true if the corresponding bit in the type word is set.

Impact paramter of the track in (r-phi). Information is stored in the first TrackState of this Track,

See Also

TrackState.

Implements **EVENT::Track** (p. 265).

7.198.3.5 float IMPL::TrackImpl::getOmega ( ) const [virtual]

Omega is the signed curvature of the track in [1/mm].

The sign is that of the particle's charge. Information is stored in the first TrackState of this Track,

See Also

TrackState.

Implements **EVENT::Track** (p. 265).

7.198.3.6 float IMPL::TrackImpl::getPhi ( ) const [virtual]

Phi of the track at reference point.

Information is stored in the first TrackState of this Track,

See Also

TrackState.

Implements **EVENT::Track** (p. 265).

7.198.3.7 float IMPL::TrackImpl::getRadiusOfInnermostHit ( ) const [virtual]

The radius of the innermost hit that has been used in the track fit.

Helps to detect V0 tracks with small impact paramters or haevy mesons. To be used as convenient information or if hits are not stored in the data set, e.g. DST or FastMC.

Implements **EVENT::Track** (p. 266).

7.198.3.8 const float * IMPL::TrackImpl::getReferencePoint ( ) const [virtual]

Reference point of the track parameters.

The default for the reference point is the point of closest approach. Information is stored in the first TrackState of this Track,

See Also

TrackState.

Implements **EVENT::Track** (p. 266).

References IMPL::TrackStateImpl::getReferencePoint().

7.198.3.9 const IntVec & IMPL::TrackImpl::getSubdetectorHitNumbers ( ) const [virtual]

A vector that holds the number of hits in particular subdetectors.

The mapping of indices to subdetectors is implementation dependent. To be used as convenient information or if hits are not stored in the data set, e.g. DST or FastMC. TODO: Provide way to store mapping in event/run header.

Implements **EVENT::Track** (p. 266).

7.198.3.10 float IMPL::TrackImpl::getTanLambda ( ) const [virtual]

Lambda is the dip angle of the track in r-z at the reference point.

Information is stored in the first TrackState of this Track,

See Also

TrackState.

Implements **EVENT::Track** (p. 266).

7.198.3.11 const TrackState ∗ IMPL::TrackImpl::getTrackState (int *location*) const [virtual]

Returns track state for the given location - or NULL if not found.

See Also

TrackState. location can be set to: AtIP, AtFirstHit, AtLastHit, AtCalorimeter, AtVertex, AtOther

Implements **EVENT::Track** (p. 266).

7.198.3.12 const TrackStateVec & IMPL::TrackImpl::getTrackStates ( ) const [virtual]

Returns track states associtated with this track.

See Also

TrackState.

Implements **EVENT::Track** (p. 267).

7.198.3.13 int IMPL::TrackImpl::getType ( ) const [virtual]

Type of track, i.e.

the subdetector(s) used to derive the track paramters, e.g. "TPC", "COMB", etc. In order to save disc space working groups should try and define three to four letter acronyms for their tracking detectors.Flagword that defines the type of track. Bits 0-15 can be used to denote the subdetectors that have contributed hits used in the track fit. The definition of the hits has to be done elsewhere, e.g. in the run header. Before LCIO 2.0 bit 31 was used to encode isReferencePointPCA (now deprecated).

Implements **EVENT::Track** (p. 267).

7.198.3.14 float IMPL::TrackImpl::getZ0 ( ) const [virtual]

Impact paramter of the track in (r-z).

Information is stored in the first TrackState of this Track,

See Also

TrackState.

Implements **EVENT::Track** (p. 267).

7.198.3.15 IntVec & IMPL::TrackImpl::subdetectorHitNumbers ( ) [virtual]

Allows modification of the subdetectorHitNumbers, e.g.

track->**subdetectorHitNumbers()** (p. 303).resize(5) ;

track->**subdetectorHitNumbers()** (p. 303)[4] = 42 ;

Referenced by SIO::SIOTrackHandler::read().

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

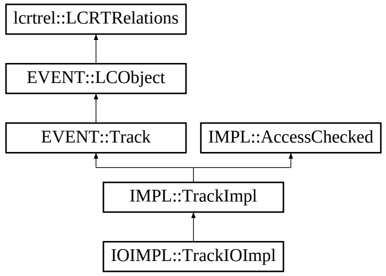
- TrackImpl.h
- TrackImpl.cc

7.199 IOIMPL::TrackIOImpl Class Reference

Adding stuff needed for io (friend declarations, etc.)

#include <IOIMPL/TrackIOImpl.h>

Inheritance diagram for IOIMPL::TrackIOImpl:



Friends

- class **SIO::SIOTrackHandler**

Additional Inherited Members

7.199.1 Detailed Description

Adding stuff needed for io (friend declarations, etc.)

Author

gaede

Version

Mar 15, 2004

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

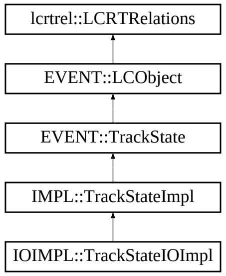
- TrackIOImpl.h

7.200 EVENT::TrackState Class Reference

The **LCIO** (p. 115) **TrackState** (p. 304) class.

#include <pre-generated/EVENT/TrackState.h>

Inheritance diagram for EVENT::TrackState:



Public Types

- typedef **TrackState lcobject_type**

  *Useful typedef for template programming with **LCIO** (p. 115).*

Public Member Functions

- virtual ∼**TrackState** ()

  Destructor.
- virtual int **getLocation** () const =0

  The location of the track state.
- virtual float **getD0** () const =0

  Impact paramter of the track in (r-phi).
- virtual float **getPhi** () const =0

  Phi of the track at the reference point.
- virtual float **getOmega** () const =0

  Omega is the signed curvature of the track in [1/mm].
- virtual float **getZ0** () const =0

  Impact paramter of the track in (r-z).
- virtual float **getTanLambda** () const =0

  Lambda is the dip angle of the track in r-z at the reference point.
- virtual const **FloatVec** & **getCovMatrix** () const =0

  Covariance matrix of the track parameters.
- virtual const float ∗ **getReferencePoint** () const =0

  Reference point of the track parameters, e.g.

Static Public Attributes

- static const int **AtOther** = 0
- static const int **AtIP** = 1
- static const int **AtFirstHit** = 2
- static const int **AtLastHit** = 3
- static const int **AtCalorimeter** = 4
- static const int **AtVertex** = 5
- static const int **LastLocation** = AtVertex

Additional Inherited Members

7.200.1 Detailed Description

The **LCIO** (p. 115) **TrackState** (p. 304) class.

Author

gaede, engels

Version

$Id:$

7.200.2 Member Function Documentation

7.200.2.1 virtual const FloatVec& EVENT::TrackState::getCovMatrix ( ) const [pure virtual]

Covariance matrix of the track parameters.

Stored as lower triangle matrix where the order of parameters is: d0, phi, omega, z0, tan(lambda). So we have cov(d0,d0), cov(phi, d0), cov(phi, phi), ...

Implemented in **IMPL::TrackStateImpl** (p. 309).

7.200.2.2 virtual int EVENT::TrackState::getLocation ( ) const [pure virtual]

The location of the track state.

Location can be set to: TrackState::AtIP, TrackState::AtFirstHit, TrackState::AtLastHit, TrackState::AtCalorimeter, TrackState::AtVertex, TrackState::AtOther

Implemented in **IMPL::TrackStateImpl** (p. 309).

Referenced by IMPL::TrackStateImpl::TrackStateImpl().

7.200.2.3 virtual float EVENT::TrackState::getOmega ( ) const [pure virtual]

Omega is the signed curvature of the track in [1/mm].

The sign is that of the particle's charge.

Implemented in **IMPL::TrackStateImpl** (p. 309).

7.200.2.4 virtual float EVENT::TrackState::getPhi ( ) const [pure virtual]

Phi of the track at the reference point.

See Also

getReferencePoint (p. 306)

Implemented in **IMPL::TrackStateImpl** (p. 308).

7.200.2.5 virtual const float∗ EVENT::TrackState::getReferencePoint ( ) const [pure virtual]

Reference point of the track parameters, e.g.

the origin at the IP, or the position of the first/last hits or the entry point into the calorimeter.

Implemented in **IMPL::TrackStateImpl** (p. 309).

Referenced by IMPL::TrackStateImpl::TrackStateImpl().

7.200.2.6 virtual float EVENT::TrackState::getTanLambda ( ) const [pure virtual]

Lambda is the dip angle of the track in r-z at the reference point.

See Also

getReferencePoint (p. 306)

Implemented in **IMPL::TrackStateImpl** (p. 308).

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

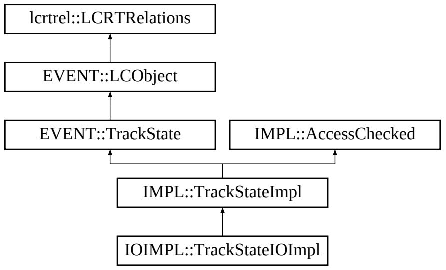
- TrackState.h
- TrackStateImpl.cc

7.201 IMPL::TrackStateImpl Class Reference

Implementation of the LCIO TrackState class.

`#include <IMPL/TrackStateImpl.h>`

Inheritance diagram for IMPL::TrackStateImpl:



Public Member Functions

- **TrackStateImpl** ()
  Default constructor, initializes values to 0.
- **TrackStateImpl** (int location, float d0, float phi, float omega, float z0, float tanLambda, const float ∗covMatrix, const float ∗reference)
- **TrackStateImpl** (int location, float d0, float phi, float omega, float z0, float tanLambda, const **EVENT::FloatVec** &covMatrix, const float ∗reference)
- **TrackStateImpl** (const **EVENT::TrackState** &p)
  Copy constructor which takes as an argument an ***EVENT::TrackState*** *(p. 304) reference.*
- virtual ∼**TrackStateImpl** ()
  Destructor.
- virtual int **id** () const
  Returns an object id for internal (debugging) use in LCIO.
- virtual int **getLocation** () const

  The location of the track state.
- virtual float **getD0** () const
  Impact paramter of the track in (r-phi).
- virtual float **getPhi** () const
  Phi of the track at reference point.
- virtual float **getOmega** () const
  Omega is the signed curvature of the track in [1/mm].
- virtual float **getZ0** () const
  Impact paramter of the track in (r-z).
- virtual float **getTanLambda** () const
  Lambda is the dip angle of the track in r-z at the reference point.
- virtual const **EVENT::FloatVec** & **getCovMatrix** () const
  Covariance matrix of the track parameters.
- virtual const float ∗ **getReferencePoint** () const
  Reference point of the track parameters, e.g.
- virtual void **setLocation** (int location)
- virtual void **setD0** (float d0)
- virtual void **setPhi** (float phi)
- virtual void **setOmega** (float omega)
- virtual void **setZ0** (float z0)
- virtual void **setTanLambda** (float tanLambda)
- virtual void **setCovMatrix** (const float ∗cov)
- virtual void **setCovMatrix** (const **EVENT::FloatVec** &cov)
- virtual void **setReferencePoint** (const float ∗rPnt)

Protected Attributes

- int **_location**
- float **_d0**
- float **_phi**
- float **_omega**
- float **_z0**
- float **_tanLambda**
- **EVENT::FloatVec _covMatrix**
- float **_reference** [TRKSTATENREFSIZE]

Additional Inherited Members

7.201.1 Detailed Description

Implementation of the LCIO TrackState class.

See Also

TrackState

Author

gaede, engels

Version

$Id:$

7.201.2 Member Function Documentation

7.201.2.1 const FloatVec & IMPL::TrackStateImpl::getCovMatrix ( ) const [virtual]

Covariance matrix of the track parameters.

Stored as lower triangle matrix where the order of parameters is: d0, phi, omega, z0, tan(lambda). So we have cov(d0,d0), cov(phi, d0), cov(phi, phi), ...

Implements **EVENT::TrackState** (p. 306).

Referenced by IMPL::TrackImpl::getCovMatrix().

7.201.2.2 int IMPL::TrackStateImpl::getLocation ( ) const [virtual]

The location of the track state.

Location can be set to: AtIP, AtFirstHit, AtLastHit, AtCalorimeter, AtVertex, AtOther

Implements **EVENT::TrackState** (p. 306).

7.201.2.3 float IMPL::TrackStateImpl::getOmega ( ) const [virtual]

Omega is the signed curvature of the track in [1/mm].

The sign is that of the particle's charge.

Implements **EVENT::TrackState** (p. 306).

7.201.2.4 const float ∗ IMPL::TrackStateImpl::getReferencePoint ( ) const [virtual]

Reference point of the track parameters, e.g.

the origin at the IP, or the position of the first/last hits or the entry point into the calorimeter.

Implements **EVENT::TrackState** (p. 306).

Referenced by IMPL::TrackImpl::getReferencePoint().

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

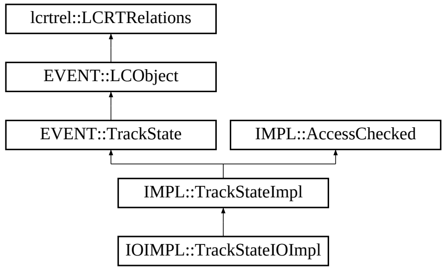
- TrackStateImpl.h
- TrackStateImpl.cc

7.202 IOIMPL::TrackStateIOImpl Class Reference

Adding stuff needed for io (friend declarations, etc.)

`#include <IOIMPL/TrackStateIOImpl.h>`

Inheritance diagram for IOIMPL::TrackStateIOImpl:



Friends

- class **SIO::SIOTrackHandler**

Additional Inherited Members

7.202.1 Detailed Description

Adding stuff needed for io (friend declarations, etc.)

Author

gaede, engels

Version

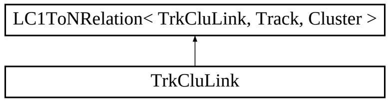
Mar 15, 2004

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

- TrackStateIOImpl.h

7.203 TrkCluLink Struct Reference

Inheritance diagram for TrkCluLink:



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

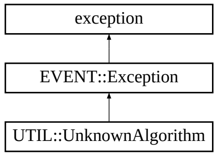
- lcrtrelation.cc

7.204 UTIL::UnknownAlgorithm Class Reference

Exception for unknown algorithms.

#include <UTIL/PIDHandler.h>

Inheritance diagram for UTIL::UnknownAlgorithm:



Public Member Functions

- **UnknownAlgorithm** (std::string text)

Additional Inherited Members

7.204.1 Detailed Description

Exception for unknown algorithms.

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

- PIDHandler.h

7.205 SIO::SIORecords::Unpack Class Reference

Helper class that sets the unpack flag of known **SIO** (p. 30) records as specified by the constructor.

#include <SIO/LCSIO.h>

Public Member Functions

- **Unpack** (unsigned recordFlag)

Static Public Attributes

- static const unsigned **Event** = 0x0001 << SIORecords::Event
- static const unsigned **Header** = 0x0001 << SIORecords::Header
- static const unsigned **Run** = 0x0001 << SIORecords::Run
- static const unsigned **Access** = 0x0001 << SIORecords::Access
- static const unsigned **Index** = 0x0001 << SIORecords::Index
- static const unsigned **All** = 0xFFFFFFFF

Protected Attributes

- bool **_flags** [NumberOfRecords]

7.205.1 Detailed Description

Helper class that sets the unpack flag of known **SIO** (p. 30) records as specified by the constructor.

After an object of this class goes out of scope the original setting is restored. This is needed to have multiple read streams (event overlay).

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

- LCSIO.h
- LCSIO.cc

7.206 UserClass Struct Reference

Public Attributes

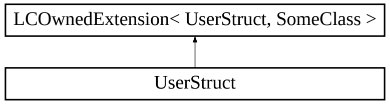
- int **someInt**
- float **aFloat**

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

- lcrtrelation.cc

7.207 UserStruct Struct Reference

Inheritance diagram for UserStruct:



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

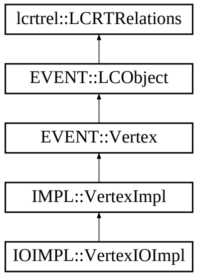
- test_RTextension.cc

7.208 EVENT::Vertex Class Reference

The **Vertex** (p. 312) class for **LCIO** (p. 115).

#include <pre-generated/EVENT/Vertex.h>

Inheritance diagram for EVENT::Vertex:



Public Types

- typedef **Vertex lcobject_type**

  Useful typedef for template programming with ***LCIO*** *(p. 115).*

Public Member Functions

- virtual ∼**Vertex** ()

  Destructor.
- virtual bool **isPrimary** () const =0

  Checks if the ***Vertex*** *(p. 312) is the primary vertex of the event.*
- virtual const std::string & **getAlgorithmType** () const =0

  Type code for the algorithm that has been used to create the vertex - check/set the collection parameters AlgorithmName and AlgorithmType.
- virtual float **getChi2** () const =0

  Chi squared of the vertex fit.
- virtual float **getProbability** () const =0

  Probability of the vertex fit.
- virtual const float ∗ **getPosition** () const =0

  Position of the vertex.
- virtual const **FloatVec** & **getCovMatrix** () const =0

  Covariance matrix of the position (stored as lower triangle matrix, i.e.
- virtual const **FloatVec** & **getParameters** () const =0

  Additional parameters related to this vertex - check/set the collection parameter "VertexParameterNames" for the parameters' meaning.
- virtual **ReconstructedParticle** ∗ **getAssociatedParticle** () const =0

  Returns Reconstructed Particle associated to the ***Vertex*** *(p. 312).*

Additional Inherited Members

7.208.1 Detailed Description

The **Vertex** (p. 312) class for **LCIO** (p. 115).

UNDER DEVELOPMENT!

see: here

UNDER DEVELOPMENT!

Author

gaede, engels

Version

Id:

Vertex.aid,v 1.3 2006-09-21 06:10:36 gaede Exp

7.208.2 Member Function Documentation

7.208.2.1 virtual const FloatVec& EVENT::Vertex::getCovMatrix ( ) const [pure virtual]

Covariance matrix of the position (stored as lower triangle matrix, i.e.

cov(xx),cov(y,x),cov(y,y)).

Implemented in **IMPL::VertexImpl** (p. 316).

Referenced by SIO::SIOVertexHandler::write().

7.208.2.2 virtual bool EVENT::Vertex::isPrimary ( ) const [pure virtual]

Checks if the **Vertex** (p. 312) is the primary vertex of the event.

Only one primary vertex per event is allowed

Implemented in **IMPL::VertexImpl** (p. 316).

Referenced by SIO::SIOVertexHandler::write().

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

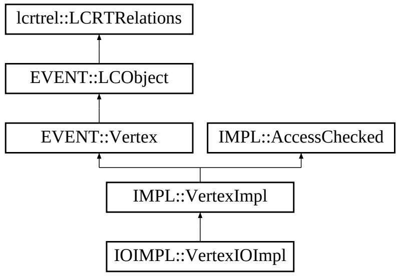
- Vertex.h

7.209 IMPL::VertexImpl Class Reference

Implementation of Vertex Class.

`#include <IMPL/VertexImpl.h>`

Inheritance diagram for IMPL::VertexImpl:



Public Member Functions

- **VertexImpl** ()

  Default constructor, initializes values to 0.
- virtual int **id** () const

  Returns an object id for internal (debugging) use in LCIO.
- virtual bool **isPrimary** () const

  Checks if the Vertex is the primary vertex of the event.
- virtual const std::string & **getAlgorithmType** () const

  Type code for the algorithm that has been used to create the vertex - check/set the collection parameters AlgorithmName and AlgorithmType.
- virtual float **getChi2** () const

  Chi squared of the vertex fit.
- virtual float **getProbability** () const

  Probability of the vertex fit.
- virtual const float ∗ **getPosition** () const

  Position of the vertex.
- virtual const **EVENT::FloatVec** & **getCovMatrix** () const

  Covariance matrix of the position (stored as lower triangle matrix, i.e.
- virtual const **EVENT::FloatVec** & **getParameters** () const

  Additional parameters related to this vertex - check/set the collection parameter "VertexParameterNames" for the parameters' meaning.
- virtual
  EVENT::ReconstructedParticle ∗ **getAssociatedParticle** () const

  Returns Reconstructed Particle associated to the Vertex.
- void **setPrimary** (bool primary)
- void **setAlgorithmType** (std::string type)
- void **setChi2** (float chi2)
- void **setProbability** (float probability)
- void **setPosition** (const float vpos[3])
- void **setPosition** (float px, float py, float pz)
- void **setCovMatrix** (const float ∗cov)
- void **setCovMatrix** (const **EVENT::FloatVec** &)
- void **setAssociatedParticle** (**EVENT::ReconstructedParticle** ∗aP)
- void **addParameter** (float p)

Protected Attributes

- int **_primary**
- std::string **_type**
- float **_chi2**
- float **_probability**
- float **_vpos** [3]
- **EVENT::FloatVec _cov**
- **EVENT::FloatVec _par**
- **EVENT::ReconstructedParticle ∗ _aParticle**

Additional Inherited Members

7.209.1 Detailed Description

Implementation of Vertex Class.

See Also

Vertex

Author

gaede, engels

Version

Aug 09, 2006

7.209.2 Member Function Documentation

7.209.2.1 const EVENT::FloatVec & IMPL::VertexImpl::getCovMatrix ( ) const [virtual]

Covariance matrix of the position (stored as lower triangle matrix, i.e.

cov(xx),cov(y,x),cov(y,y)).

Implements **EVENT::Vertex** (p. 314).

7.209.2.2 bool IMPL::VertexImpl::isPrimary ( ) const [virtual]

Checks if the Vertex is the primary vertex of the event.

Only one primary vertex per event is allowed

Implements **EVENT::Vertex** (p. 314).

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

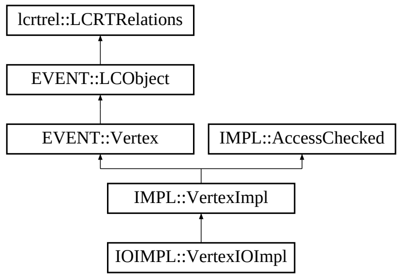
- VertexImpl.h
- VertexImpl.cc

7.210 IOIMPL::VertexIOImpl Class Reference

Adding stuff needed for io (friend declarations, etc.)

`#include <IOIMPL/VertexIOImpl.h>`

Inheritance diagram for IOIMPL::VertexIOImpl:



Friends

- class **SIO::SIOVertexHandler**

Additional Inherited Members

7.210.1 Detailed Description

Adding stuff needed for io (friend declarations, etc.)

Author

gaede, engels

Version

Aug 15, 2006

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

- VertexIOImpl.h