Reference Manual

Generated by Doxygen 1.8.3

Mon Feb 11 2013 14:43:03

Contents

Chapter 1

Namespace Index

1.1 Namespace List

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

Chapter 2

Hierarchical Index

2.1 Class Hierarchy

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

Chapter 3

Class Index

3.1 Class List

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

Chapter 4

Namespace Documentation

4.1 ini Namespace Reference

The namespace of the INI configuration file parser.

Classes

- class Value
- class ParseException

    The base class for all exceptions that are thrown by the INI-parser.
- class UnexpectedCharacter

    The exception that is thrown when the parser encounters an unexpected character.
- class DuplicateSection

    The exception that is thrown by the parser when it encounters a section that has the same name as a previously parsed section.
- class DuplicateEntry

    The exception that is thrown when the parser encounters an entry that has the same key as a previously parsed key in the same section.
- class NonexistentEntry

    The execption that is thrown when the value of a nonexistent entry is requested.
- class IncompatibleConversion

    The execption that is thrown when the value of an entry in an ini-configuration cannot be converted to the requested value.
- class Entry

    The class that represents an entry in the section of an INI configuration.
- class Section

    The type that is used to represent sections that are stored in the configuration file.
- class Configuration

    The type in which a configuration file is stored.

Typedefs

- typedef std::vector< int > IntTuple

    The type that is used to store int tuples (list of ints).
- typedef std::vector< double > DoubleTuple

    The type that is used to store double tuples (list of doubles).
- typedef std::map< std::string,
  Value ∗ > ValueMap

The type of the map in which the values are stored.

- typedef ValueMap::iterator ValueIter

  The type of the iterator for iterating over a ValueMap.
- typedef ValueMap::const_iterator ConstValueIter

  The type of the iterator for iterating over a constant ValueMap.

Functions

- std::istream & operator>> (std::istream &input_stream, Configuration &configuration)

  Convenience operator for reading configurations from an input stream.
- std::ostream & operator<< (std::ostream &output_stream, const Configuration &configuration)

  Convenience operator for writing configurations to an output stream.

4.1.1 Detailed Description

The namespace of the INI configuration file parser.

4.1.2 Function Documentation

4.1.2.1 std::ostream & ini::operator<< (std::ostream & *output_stream*, const Configuration & *configuration*)

Convenience operator for writing configurations to an output stream.

This operator prints the configuration to the output stream using the Configuration::print method.

Parameters

output_stream	The output stream to which the configuration is written.
configuration	The Configuration object in which the parsed configuration is stored.

Returns

A reference to the output stream.

Definition at line 1610 of file ini_configuration.cc.

4.1.2.2 std::istream & ini::operator>> (std::istream & *input_stream*, Configuration & *configuration*)

Convenience operator for reading configurations from an input stream.

This operator reads the configuration from the input stream using the Configuration::parse method.

Parameters

input_stream	The input stream from which the configuration is read.
configuration	The Configuration object in which the parsed configuration is stored.

Returns

A reference to the input stream.

Definition at line 1603 of file ini_configuration.cc.

Chapter 5

Class Documentation

5.1 ini::Configuration Class Reference

The type in which a configuration file is stored.

`#include <ini_configuration.hh>`

Public Member Functions

- Configuration ()

  Constructs a new (empty) configuration.
- Configuration (std::istream &input_stream)

  Constructs a new Configuration by parsing the content from a stream.
- ∼Configuration ()

  Destructs a Configuration and frees all entries stored in it.
- Section operator[] (const std::string &key) const

  Retrieves a section from the configuration file given its key.
- void parse (std::istream &input_stream)

  Reads a configuration file from a stream.
- void print (std::ostream &output_stream) const

  Formats the contents of the Configuration to text and prints it to an output stream.

5.1.1 Detailed Description

The type in which a configuration file is stored.

Definition at line 918 of file ini_configuration.hh.

5.1.2 Constructor & Destructor Documentation

5.1.2.1 ini::Configuration::Configuration (std::istream & *input_stream*)

Constructs a new Configuration by parsing the content from a stream.

Parameters

input_stream	The stream from which the content is parsed.

Definition at line 1509 of file ini_configuration.cc.

5.1.3 Member Function Documentation

5.1.3.1 Section ini::Configuration::operator[] (const std::string & *key*) const

Retrieves a section from the configuration file given its key.

If the requested section does not exist, a section containing no values is returned.

Parameters

key	The name of the requested section.

Returns

A reference to the requested section.

Definition at line 1539 of file ini_configuration.cc.

5.1.3.2 void ini::Configuration::parse (std::istream & *input_stream*)

Reads a configuration file from a stream.

Parameters

input_stream	The input stream from which the configuration is read.

Definition at line 1553 of file ini_configuration.cc.

5.1.3.3 void ini::Configuration::print (std::ostream & *output_stream*) const

Formats the contents of the Configuration to text and prints it to an output stream.

Parameters

output_stream	The output stream to which the output is written.

Definition at line 1574 of file ini_configuration.cc.

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

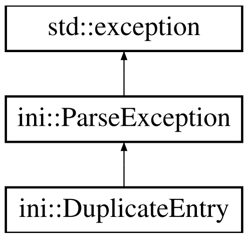
- /Users/bartsas/Courses/Graphics/SVN/code/cxx/ini_configuration/ini_configuration.hh
- /Users/bartsas/Courses/Graphics/SVN/code/cxx/ini_configuration/ini_configuration.cc

5.2 ini::DuplicateEntry Class Reference

The exception that is thrown when the parser encounters an entry that has the same key as a previously parsed key in the same section.

`#include <ini_configuration.hh>`

Inheritance diagram for ini::DuplicateEntry:



Public Member Functions

- DuplicateEntry (const std::string &section_init, const std::string &key_init) throw ()

    Constructs a new DuplicateEntry instance.
- DuplicateEntry (const DuplicateEntry &original) throw ()

    Constructs a new DuplicateEntry instance by copying another one.
- virtual ∼DuplicateEntry () throw ()

    Destructs a DuplicateEntry.
- DuplicateEntry & operator= (const DuplicateEntry &original) throw ()

    Copies a DuplicateEntry exception.
- virtual const char ∗ what () const throw ()

    Returns a description of the error hat occurred.

Additional Inherited Members

5.2.1 Detailed Description

The exception that is thrown when the parser encounters an entry that has the same key as a previously parsed key in the same section.

Definition at line 198 of file ini_configuration.hh.

5.2.2 Constructor & Destructor Documentation

5.2.2.1 ini::DuplicateEntry::DuplicateEntry (const std::string & *section_init*, const std::string & *key_init*) throw ()

Constructs a new DuplicateEntry instance.

Parameters

section_init	The name of the section that contains the duplicate entry.
key_init	The name of the duplicate entry.

Definition at line 168 of file ini_configuration.cc.

5.2.2.2 ini::DuplicateEntry::DuplicateEntry (const **DuplicateEntry** & *original*) throw ()

Constructs a new DuplicateEntry instance by copying another one.

Parameters

original	The instance that is copied.

Definition at line 182 of file ini_configuration.cc.

5.2.3 Member Function Documentation

5.2.3.1 DuplicateEntry & ini::DuplicateEntry::operator= (const DuplicateEntry & *original*) throw ()

Copies a DuplicateEntry exception.

Parameters

original	The instance that is copied.

Returns

A reference to this instance.

Definition at line 196 of file ini_configuration.cc.

5.2.3.2 const char ∗ ini::DuplicateEntry::what ( ) const throw () [virtual]

Returns a description of the error hat occurred.

Returns

A description of the error hat occurred.

Implements ini::ParseException.

Definition at line 207 of file ini_configuration.cc.

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

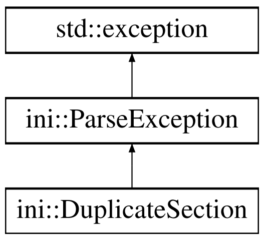
- /Users/bartsas/Courses/Graphics/SVN/code/cxx/ini_configuration/ini_configuration.hh
- /Users/bartsas/Courses/Graphics/SVN/code/cxx/ini_configuration/ini_configuration.cc

5.3 ini::DuplicateSection Class Reference

The exception that is thrown by the parser when it encounters a section that has the same name as a previously parsed section.

`#include <ini_configuration.hh>`

Inheritance diagram for ini::DuplicateSection:



Public Member Functions

- DuplicateSection (const std::string &name_init) throw ()

  Constructs a new DuplicateSection instance.
- DuplicateSection (const DuplicateSection &original) throw ()

Constructs a new DuplicateSection instance by copying another one.

- virtual ∼DuplicateSection () throw ()

  Destructs a DuplicateSection.
- DuplicateSection & operator= (const DuplicateSection &original) throw ()

  Copies a DuplicateSection.
- virtual const char ∗ what () const throw ()

  Returns a description of the error hat occurred.

Additional Inherited Members

5.3.1 Detailed Description

The exception that is thrown by the parser when it encounters a section that has the same name as a previously parsed section.

Definition at line 143 of file ini_configuration.hh.

5.3.2 Constructor & Destructor Documentation

5.3.2.1 ini::DuplicateSection::DuplicateSection (const std::string & *name_init*) throw ()

Constructs a new DuplicateSection instance.

Parameters

name_init	The name of the duplicate section.

Definition at line 128 of file ini_configuration.cc.

5.3.2.2 ini::DuplicateSection::DuplicateSection (const **DuplicateSection** & *original*) throw ()

Constructs a new DuplicateSection instance by copying another one.

Parameters

original	The instance that is copied.

Definition at line 138 of file ini_configuration.cc.

5.3.3 Member Function Documentation

5.3.3.1 **DuplicateSection** & ini::DuplicateSection::operator= (const **DuplicateSection** & *original*) throw ()

Copies a DuplicateSection.

Parameters

original	The instance that is copied.

Returns

A reference to this instance.

Definition at line 151 of file ini_configuration.cc.

5.3.3.2 const char ∗ ini::DuplicateSection::what ( ) const throw () [virtual]

Returns a description of the error hat occurred.

Returns

A description of the error hat occurred.

Implements ini::ParseException.

Definition at line 161 of file ini_configuration.cc.

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

- /Users/bartsas/Courses/Graphics/SVN/code/cxx/ini_configuration/ini_configuration.hh
- /Users/bartsas/Courses/Graphics/SVN/code/cxx/ini_configuration/ini_configuration.cc

5.4 ini::Entry Class Reference

The class that represents an entry in the section of an INI configuration.

`#include <ini_configuration.hh>`

Public Member Functions

- Entry (const std::string &section_name_init, const std::string &entry_name_init, const Value ∗const value_-ptr_init)

  Constructs a new entry given the name of the section it belongs to and its value.
- Entry (const Entry &original)

  Constructs an entry by copying another one.
- ∼Entry ()

  Destructs an entry.
- Entry & operator= (const Entry &original)

  Copies an entry.
- const std::string & get_section_name () const

  Returns the name of the section to which this entry belongs.
- const std::string & get_entry_name () const

  Returns the name of this entry.
- bool exists () const

  Checks whether this entry exists in the configuration or not.
- bool as_int_if_exists (int &ret_val) const

  Returns the value as an int.
- bool as_double_if_exists (double &ret_val) const

  Returns the value as a double.
- bool as_string_if_exists (std::string &ret_val) const

  Returns the value as a string.
- bool as_bool_if_exists (bool &ret_val) const

  Returns the value as a bool.
- bool as_int_tuple_if_exists (IntTuple &ret_val) const

  Returns the value as an int tuple.
- bool as_double_tuple_if_exists (DoubleTuple &ret_val) const

  Returns the value as a double tuple.
- int as_int_or_die () const

Returns the value as an int.

- double as_double_or_die () const

  Returns the value as a double.
- std::string as_string_or_die () const

  Returns the value as a string.
- bool as_bool_or_die () const

  Returns the value as a bool.
- IntTuple as_int_tuple_or_die () const

  Returns the value as an int tuple.
- DoubleTuple as_double_tuple_or_die () const

  Returns the value as a double tuple.
- int as_int_or_default (const int def_val) const

  Returns the value as an int.
- double as_double_or_default (const double def_val) const

  Returns the value as a double.
- std::string as_string_or_default (const std::string &def_val) const

  Returns the value as a string.
- bool as_bool_or_default (const bool def_val) const

  Returns the value as a bool.
- IntTuple as_int_tuple_or_default (const IntTuple &def_val) const

  Returns the value as an int tuple.
- DoubleTuple as_double_tuple_or_default (const DoubleTuple &def_val) const

  Returns the value as a double tuple.
- operator int () const

  An alias for as_int_or_die.
- operator double () const

  An alias for as_double_or_die.
- operator std::string () const

  An alias for as_string_or_die.
- operator bool () const

  An alias for as_bool_or_die.
- operator IntTuple () const

  An alias for as_int_tuple_or_die.
- operator DoubleTuple () const

  An alias for as_int_or_die.
- int operator|| (const int def_val) const

  An alias for as_int_or_default.
- double operator|| (const double def_val) const

  An alias for as_double_or_default.
- std::string operator|| (const std::string &def_val) const

  An alias for as_string_or_default.
- bool operator|| (const bool def_val) const

  An alias for as_bool_or_default.
- IntTuple operator|| (const IntTuple &def_val) const

  An alias for as_int_tuple_or_default.
- DoubleTuple operator|| (const DoubleTuple &def_val) const

  An alias for as_double_tuple_or_default.

5.4.1 Detailed Description

The class that represents an entry in the section of an INI configuration.

Definition at line 410 of file ini_configuration.hh.

5.4.2 Constructor & Destructor Documentation

5.4.2.1 ini::Entry::Entry (const std::string & *section_name_init,* const std::string & *entry_name_init,* const **Value** ∗const *value_ptr_init*)

Constructs a new entry given the name of the section it belongs to and its value.

Parameters

section_name_-init	The name of the section to which this entry belongs.
entry_name_init	The name of this entry.
value_ptr_init	A pointer to the value of this entry.

Definition at line 1176 of file ini_configuration.cc.

5.4.2.2 ini::Entry::Entry (const **Entry** & *original*)

Constructs an entry by copying another one.

Parameters

original	The section whose values are copied.

Definition at line 1186 of file ini_configuration.cc.

5.4.3 Member Function Documentation

5.4.3.1 bool ini::Entry::as_bool_if_exists (bool & *ret_val*) const

Returns the value as a bool.

If the entry exists and can be represented as a bool, the value is passed to the caller through the parameter and `true` is returned. If the entry exists but is not representable as a bool, an IncompatibleConversion exception is thrown. If the entry does not exist, `false` is returned and the value of the parameter is not changed.

Parameters

ret_val	The parameter through which the value is returned.

Returns

`true` if the value exists, `false` otherwise.

Definition at line 1238 of file ini_configuration.cc.

5.4.3.2 bool ini::Entry::as_bool_or_default (const bool *def_val*) const

Returns the value as a bool.

If the entry exists and can be represented as a bool, it is returned. If the value is not representable as a bool, an IncompatibleConversion exception is thrown. If the entry does not exist, a default value is returned.

Parameters

def_val	The default value that is returned if the value does not exist.

Returns

The value as a bool or the default value if the value does not exist.

Definition at line 1361 of file ini_configuration.cc.

5.4.3.3 bool ini::Entry::as_bool_or_die ( ) const

Returns the value as a bool.

If the entry exists and can be represented as a bool, it is returned. If the value is not representable as a bool, an IncompatibleConversion exception is thrown. If the entry does not exist, a NonexistentEntry exception is thrown.

Returns

The value as a bool.

Definition at line 1289 of file ini_configuration.cc.

5.4.3.4 bool ini::Entry::as_double_if_exists (double & *ret_val*) const

Returns the value as a double.

If the entry exists and can be represented as a double, the value is passed to the caller through the parameter and `true` is returned. If the entry exists but is not representable as a double, an IncompatibleConversion exception is thrown. If the entry does not exist, `false` is returned and the value of the parameter is not changed.

Parameters

ret_val	The parameter through which the value is returned.

Returns

`true` if the value exists, `false` otherwise.

Definition at line 1228 of file ini_configuration.cc.

5.4.3.5 double ini::Entry::as_double_or_default (const double *def_val*) const

Returns the value as a double.

If the entry exists and can be represented as a double, it is returned. If the value is not representable as a double, an IncompatibleConversion exception is thrown. If the entry does not exist, a default value is returned.

Parameters

def_val	The default value that is returned if the value does not exist.

Returns

The value as a double or the default value if the value does not exist.

Definition at line 1337 of file ini_configuration.cc.

5.4.3.6 double ini::Entry::as_double_or_die ( ) const

Returns the value as a double.

If the entry exists and can be represented as a double, it is returned. If the value is not representable as a double, an IncompatibleConversion exception is thrown. If the entry does not exist, a NonexistentEntry exception is thrown.

Returns

The value as a double.

Definition at line 1265 of file ini_configuration.cc.

5.4.3.7 bool ini::Entry::as_double_tuple_if_exists (DoubleTuple & *ret_val*) const

Returns the value as a double tuple.

If the entry exists and can be represented as a double tuple, the value is passed to the caller through the parameter and `true` is returned. If the value exists but is not representable as a double tuple, an IncompatibleConversion exception is thrown. If the entry does not exist, `false` is returned and the value of the parameter is not changed.

Parameters

ret_val	The parameter through which the value is returned.

Returns

`true` if the value exists, `false` otherwise.

Definition at line 1248 of file ini_configuration.cc.

5.4.3.8 DoubleTuple ini::Entry::as_double_tuple_or_default (const DoubleTuple & *def_val*) const

Returns the value as a double tuple.

If the entry exists and can be represented as a double tuple, it is returned. If the value is not representable as a double tuple, an IncompatibleConversion exception is thrown. If the entry does not exist, a default value is returned.

Parameters

def_val	The default value that is returned if the value does not exist.

Returns

The value as a double tuple or the default value if the value does not exist.

Definition at line 1385 of file ini_configuration.cc.

5.4.3.9 DoubleTuple ini::Entry::as_double_tuple_or_die ( ) const

Returns the value as a double tuple.

If the entry exists and can be represented as a double tuple, it is returned. If the value is not representable as a double tuple, an IncompatibleConversion exception is thrown. If the entry does not exist, a NonexistentEntry exception is thrown.

Returns

The value as a double tuple.

Definition at line 1313 of file ini_configuration.cc.

5.4.3.10 bool ini::Entry::as_int_if_exists (int & *ret_val*) const

Returns the value as an int.

If the entry exists and can be represented as an int, the value is passed to the caller through the parameter and `true` is returned. If the entry exists but is not representable as an int, an IncompatibleConversion exception is thrown. If the entry does not exist, `false` is returned and the value of the parameter is not changed.

Parameters

ret_val	The parameter through which the value is returned.

Returns

`true` if the value exists, `false` otherwise.

Definition at line 1223 of file ini_configuration.cc.

5.4.3.11 int ini::Entry::as_int_or_default (const int *def_val*) const

Returns the value as an int.

If the entry exists and can be represented as an int, it is returned. If the value is not representable as an int, an IncompatibleConversion exception is thrown. If the entry does not exist, a default value is returned.

Parameters

def_val	The default value that is returned if the value does not exist.

Returns

The value as an int or the default value if the value does not exist.

Definition at line 1325 of file ini_configuration.cc.

5.4.3.12 int ini::Entry::as_int_or_die ( ) const

Returns the value as an int.

If the entry exists and can be represented as an int, it is returned. If the value is not representable as an int, an IncompatibleConversion exception is thrown. If the entry does not exist, a NonexistentEntry exception is thrown.

Returns

The value as an int.

Definition at line 1253 of file ini_configuration.cc.

5.4.3.13 bool ini::Entry::as_int_tuple_if_exists (IntTuple & *ret_val*) const

Returns the value as an int tuple.

If the entry exists and can be represented as an int tuple, the value is passed to the caller through the parameter and `true` is returned. If the value exists but is not representable as an int tuple, an IncompatibleConversion exception is thrown. If the entry does not exist, `false` is returned and the value of the parameter is not changed.

Parameters

ret_val	The parameter through which the value is returned.

Returns

`true` if the value exists, `false` otherwise.

Definition at line 1243 of file ini_configuration.cc.

5.4.3.14 IntTuple ini::Entry::as_int_tuple_or_default (const IntTuple & *def_val*) const

Returns the value as an int tuple.

If the entry exists and can be represented as an int tuple, it is returned. If the value is not representable as an int tuple, an IncompatibleConversion exception is thrown. If the entry does not exist, a default value is returned.

Parameters

def_val	The default value that is returned if the value does not exist.

Returns

The value as an int tuple or the default value if the value does not exist.

Definition at line 1373 of file ini_configuration.cc.

5.4.3.15 IntTuple ini::Entry::as_int_tuple_or_die ( ) const

Returns the value as an int tuple.

If the entry exists and can be represented as an int tuple, it is returned. If the value is not representable as an int tuple, an IncompatibleConversion exception is thrown. If the entry does not exist, a NonexistentEntry exception is thrown.

Returns

The value as an int tuple.

Definition at line 1301 of file ini_configuration.cc.

5.4.3.16 bool ini::Entry::as_string_if_exists (std::string & *ret_val*) const

Returns the value as a string.

If the entry exists and can be represented as a string, the value is passed to the caller through the parameter and `true` is returned. If the entry exists but is not representable as a string, an IncompatibleConversion exception is thrown. If the entry does not exist, `false` is returned and the value of the parameter is not changed.

Parameters

ret_val	The parameter through which the value is returned.

Returns

`true` if the value exists, `false` otherwise.

Definition at line 1233 of file ini_configuration.cc.

5.4.3.17 std::string ini::Entry::as_string_or_default (const std::string & *def_val*) const

Returns the value as a string.

If the entry exists and can be represented as a string, it is returned. If the value is not representable as a string, an IncompatibleConversion exception is thrown. If the entry does not exist, a default value is returned.

Parameters

def_val	The default value that is returned if the value does not exist.

Returns

The value as a string or the default value if the value does not exist.

Definition at line 1349 of file ini_configuration.cc.

5.4.3.18 std::string ini::Entry::as_string_or_die ( ) const

Returns the value as a string.

If the entry exists and can be represented as a string, it is returned. If the value is not representable as a string, an IncompatibleConversion exception is thrown. If the entry does not exist, a NonexistentEntry exception is thrown.

Returns

The value as a string.

Definition at line 1277 of file ini_configuration.cc.

5.4.3.19 bool ini::Entry::exists ( ) const

Checks whether this entry exists in the configuration or not.

Returns

`true` if this entry exits, `false` otherwise.

Definition at line 1218 of file ini_configuration.cc.

5.4.3.20 const std::string & ini::Entry::get_entry_name ( ) const

Returns the name of this entry.

Returns

Returns the name of this entry.

Definition at line 1213 of file ini_configuration.cc.

5.4.3.21 const std::string & ini::Entry::get_section_name ( ) const

Returns the name of the section to which this entry belongs.

Returns

The name of the section to which this entry belongs.

Definition at line 1208 of file ini_configuration.cc.

5.4.3.22 ini::Entry::operator bool ( ) const

An alias for as_bool_or_die.

This conversion operator allows a Value to be converted to a bool when it is assigned to a bool variable.

Returns

The bool value of the Value.

Definition at line 1412 of file ini_configuration.cc.

5.4.3.23 ini::Entry::operator double ( ) const

An alias for as_double_or_die.

This conversion operator allows a Value to be converted to a double when it is assigned to a double variable.

Returns

The double value of the Value.

Definition at line 1402 of file ini_configuration.cc.

5.4.3.24 ini::Entry::operator DoubleTuple ( ) const

An alias for as_int_or_die.

This conversion operator allows a Value to be converted to a double tuple when it is assigned to a double tuple variable.

Returns

The double tuple value of the Value.

Definition at line 1422 of file ini_configuration.cc.

5.4.3.25 ini::Entry::operator int ( ) const

An alias for as_int_or_die.

This conversion operator allows a Value to be converted to an int when it is assigned to an int variable.

Returns

The int value of the Value.

Definition at line 1397 of file ini_configuration.cc.

5.4.3.26 ini::Entry::operator IntTuple ( ) const

An alias for as_int_tuple_or_die.

This conversion operator allows a Value to be converted to an int tuple when it is assigned to an int tuple variable.

Returns

The int tuple value of the Value.

Definition at line 1417 of file ini_configuration.cc.

5.4.3.27 ini::Entry::operator std::string ( ) const

An alias for as_string_or_die.

This conversion operator allows a Value to be converted to a string when it is assigned to a string variable.

Returns

The string value of the Value.

Definition at line 1407 of file ini_configuration.cc.

5.4.3.28 Entry & ini::Entry::operator= (const Entry & *original*)

Copies an entry.

Parameters

original	The section whose values are copied.

Returns

A reference to this entry.

Definition at line 1199 of file ini_configuration.cc.

5.4.3.29 int ini::Entry::operator|| (const int *def_val*) const

An alias for as_int_or_default.

Parameters

def_val	The value that is returned if the requested value does not exist.

Returns

The requested value as an int or def_val if the value does not exist.

Definition at line 1427 of file ini_configuration.cc.

5.4.3.30 double ini::Entry::operator|| (const double *def_val*) const

An alias for as_double_or_default.

Parameters

def_val	The value that is returned if the requested value does not exist.

Returns

The requested value as a double or def_val if the value does not exist.

Definition at line 1432 of file ini_configuration.cc.

5.4.3.31 std::string ini::Entry::operator\|\| (const std::string & *def_val*) const

An alias for as_string_or_default.

Parameters

def_val	The value that is returned if the requested value does not exist.

Returns

The requested value as a string or def_val if the value does not exist.

Definition at line 1437 of file ini_configuration.cc.

5.4.3.32 bool ini::Entry::operator\|\| (const bool *def_val*) const

An alias for as_bool_or_default.

Parameters

def_val	The value that is returned if the requested value does not exist.

Returns

The requested value as a bool or def_val if the value does not exist.

Definition at line 1442 of file ini_configuration.cc.

5.4.3.33 IntTuple ini::Entry::operator\|\| (const IntTuple & *def_val*) const

An alias for as_int_tuple_or_default.

Parameters

def_val	The value that is returned if the requested value does not exist.

Returns

The requested value as an int tuple or def_val if the value does not exist.

Definition at line 1447 of file ini_configuration.cc.

5.4.3.34 DoubleTuple ini::Entry::operator\|\| (const DoubleTuple & *def_val*) const

An alias for as_double_tuple_or_default.

Parameters

def_val	The value that is returned if the requested value does not exist.

Returns

The requested value as a double tuple or def_val if the value does not exist.

Definition at line 1452 of file ini_configuration.cc.

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

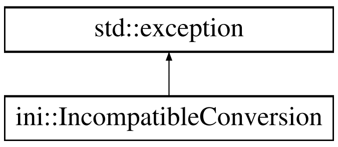
- /Users/bartsas/Courses/Graphics/SVN/code/cxx/ini_configuration/ini_configuration.hh
- /Users/bartsas/Courses/Graphics/SVN/code/cxx/ini_configuration/ini_configuration.cc

5.5 ini::IncompatibleConversion Class Reference

The execption that is thrown when the value of an entry in an ini-configuration cannot be converted to the requested value.

`#include <ini_configuration.hh>`

Inheritance diagram for ini::IncompatibleConversion:



Public Member Functions

- IncompatibleConversion (const std::string &section_name_init, const std::string &entry_name_init, const std::string &type_name_init) throw ()

  Constructs a new IncompatibleConversion exception.
- IncompatibleConversion (const IncompatibleConversion &original) throw ()

  Constructs a IncompatibleConversion exception by copying another one.
- virtual ∼IncompatibleConversion () throw ()

  Destructs a .
- IncompatibleConversion & operator= (const IncompatibleConversion &original) throw ()

  Copies an IncompatibleConversion.
- virtual const char ∗ what () const throw ()

  Returns a description of the error hat occurred.

5.5.1 Detailed Description

The execption that is thrown when the value of an entry in an ini-configuration cannot be converted to the requested value.

Definition at line 324 of file ini_configuration.hh.

5.5.2 Constructor & Destructor Documentation

5.5.2.1 ini::IncompatibleConversion::IncompatibleConversion (const std::string & *section_name_init,* const std::string & *entry_name_init,* const std::string & *type_name_init*) throw ()

Constructs a new IncompatibleConversion exception.

Parameters

section_name_-init	The name of the section of the entry.
entry_name_init	The name of the entry.
type_name_init	The name of the requested type, e.g. "int".

Definition at line 254 of file ini_configuration.cc.

5.5.2.2 ini::IncompatibleConversion::IncompatibleConversion (const IncompatibleConversion & *original*) throw ()

Constructs a IncompatibleConversion exception by copying another one.

Parameters

original	The instance that is copied.

Definition at line 271 of file ini_configuration.cc.

5.5.3 Member Function Documentation

5.5.3.1 IncompatibleConversion & ini::IncompatibleConversion::operator= (const IncompatibleConversion & *original*) throw ()

Copies an IncompatibleConversion.

Parameters

original	The instance that is copied.

Returns

A reference to this instance.

Definition at line 286 of file ini_configuration.cc.

5.5.3.2 const char * ini::IncompatibleConversion::what ( ) const throw () `[virtual]`

Returns a description of the error hat occurred.

Returns

A description of the error hat occurred.

Definition at line 298 of file ini_configuration.cc.

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

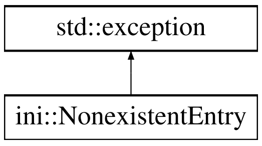
- /Users/bartsas/Courses/Graphics/SVN/code/cxx/ini_configuration/ini_configuration.hh
- /Users/bartsas/Courses/Graphics/SVN/code/cxx/ini_configuration/ini_configuration.cc

5.6 ini::NonexistentEntry Class Reference

The execption that is thrown when the value of a nonexistent entry is requested.

`#include <ini_configuration.hh>`

Inheritance diagram for ini::NonexistentEntry:



Public Member Functions

- NonexistentEntry (const std::string &section_name_init, const std::string &entry_name_init) throw ()

  Constructs a new NonexistentEntry instance.
- NonexistentEntry (const NonexistentEntry &original) throw ()

  Constructs a new NonexistentEntry instance by copying another one.
- virtual ∼NonexistentEntry () throw ()

  Destructs a NonexistentEntry.
- NonexistentEntry & operator= (const NonexistentEntry &original) throw ()

  Copies a NonexistentEntry.
- const char ∗ what () const throw ()

  Returns a description of the error hat occurred.

5.6.1 Detailed Description

The execption that is thrown when the value of a nonexistent entry is requested.

Note that this exception is only thrown when the value of an Entry is obtained; not when the entry is obtained from a Section.

Definition at line 262 of file ini_configuration.hh.

5.6.2 Constructor & Destructor Documentation

5.6.2.1 ini::NonexistentEntry::NonexistentEntry (const std::string & *section_name_init*, const std::string & *entry_name_init*) throw ()

Constructs a new NonexistentEntry instance.

Parameters

section_name_-init	The name of the section from which the nonexistent entry is obtained.
entry_name_init	The name of the nonexistent entry.

Definition at line 214 of file ini_configuration.cc.

5.6.2.2 ini::NonexistentEntry::NonexistentEntry (const **NonexistentEntry** & *original*) throw ()

Constructs a new NonexistentEntry instance by copying another one.

Parameters

original	The instance that is copied.

Definition at line 228 of file ini_configuration.cc.

5.6.3 Member Function Documentation

5.6.3.1 NonexistentEntry & ini::NonexistentEntry::operator= (const NonexistentEntry & *original*) throw ()

Copies a NonexistentEntry.

Parameters

original	The instance that is copied.

Returns

A reference to this instance.

Definition at line 240 of file ini_configuration.cc.

5.6.3.2 const char ∗ ini::NonexistentEntry::what ( ) const throw ()

Returns a description of the error hat occurred.

Returns

A description of the error hat occurred.

Definition at line 247 of file ini_configuration.cc.

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

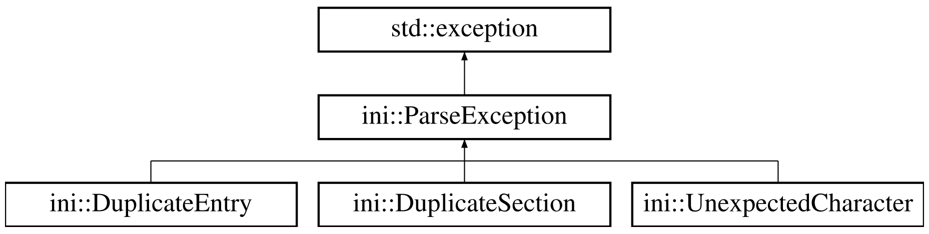
- /Users/bartsas/Courses/Graphics/SVN/code/cxx/ini_configuration/ini_configuration.hh
- /Users/bartsas/Courses/Graphics/SVN/code/cxx/ini_configuration/ini_configuration.cc

5.7 ini::ParseException Class Reference

The base class for all exceptions that are thrown by the INI-parser.

`#include <ini_configuration.hh>`

Inheritance diagram for ini::ParseException:



Public Member Functions

- virtual const char ∗ what () const =0 throw ()
  Returns a description of the error that occurred.

Protected Member Functions

- ParseException () throw ()

  Constructs a new ParseException instance.
- ParseException (const ParseException &original) throw ()

  Constructs a new ParseException instance by copying another one.
- virtual ∼ParseException () throw ()

  Destructs a ParseException.
- ParseException & operator= (const ParseException &original) throw ()

  Copies a parse exception.

5.7.1 Detailed Description

The base class for all exceptions that are thrown by the INI-parser.

Definition at line 38 of file ini_configuration.hh.

5.7.2 Constructor & Destructor Documentation

5.7.2.1 ini::ParseException::ParseException (const **ParseException** & *original*) throw () [protected]

Constructs a new ParseException instance by copying another one.

Parameters

original	The exception that is copied.

Definition at line 40 of file ini_configuration.cc.

5.7.3 Member Function Documentation

5.7.3.1 **ParseException** & ini::ParseException::operator= (const **ParseException** & *original*) throw () [protected]

Copies a parse exception.

Parameters

original	The parse exception that is copied.

Returns

A reference to this instance.

Definition at line 51 of file ini_configuration.cc.

5.7.3.2 virtual const char∗ ini::ParseException::what ( ) const throw () [pure virtual]

Returns a description of the error that occurred.

Returns

A description of the error that occurred.

Implemented in ini::DuplicateEntry, ini::DuplicateSection, and ini::UnexpectedCharacter.

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

- /Users/bartsas/Courses/Graphics/SVN/code/cxx/ini_configuration/ini_configuration.hh
- /Users/bartsas/Courses/Graphics/SVN/code/cxx/ini_configuration/ini_configuration.cc

5.8 ini::Section Class Reference

The type that is used to represent sections that are stored in the configuration file.

`#include <ini_configuration.hh>`

Public Member Functions

- Section (const std::string &section_name_init, const ValueMap ∗const values_init)

  Creates a new section.
- Section (const Section &original)

  Creates a new section by copying another one.
- ∼Section ()

  Destructs a section.
- Section & operator= (const Section &original)

  Copies another section.
- Entry operator[] (const std::string &key) const

  Looks up a entry in the section given its key and returns it.

5.8.1 Detailed Description

The type that is used to represent sections that are stored in the configuration file.

Definition at line 859 of file ini_configuration.hh.

5.8.2 Constructor & Destructor Documentation

5.8.2.1 ini::Section::Section (const std::string & *section_name_init*, const ValueMap ∗const *values_init*)

Creates a new section.

Parameters

section_name_-init	The name of the section.
values_init	An iterator to the map that stores the entries of the section.

Definition at line 1459 of file ini_configuration.cc.

5.8.2.2 ini::Section::Section (const Section & *original*)

Creates a new section by copying another one.

Parameters

original	The section that is copied.

Definition at line 1467 of file ini_configuration.cc.

5.8.3 Member Function Documentation

5.8.3.1 Section& ini::Section::operator= (const Section & *original*)

Copies another section.

Parameters

original	The section that is copied.

Returns

A reference to this section.

5.8.3.2 Entry ini::Section::operator[] (const std::string & *key*) const

Looks up a entry in the section given its key and returns it.

Parameters

key	The entry corresponding to the key.

Returns

The entry corresponding to the key or an empty entry if the requested entry does not exist.

Definition at line 1479 of file ini_configuration.cc.

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

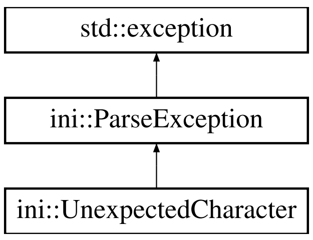
- /Users/bartsas/Courses/Graphics/SVN/code/cxx/ini_configuration/ini_configuration.hh
- /Users/bartsas/Courses/Graphics/SVN/code/cxx/ini_configuration/ini_configuration.cc

5.9 ini::UnexpectedCharacter Class Reference

The exception that is thrown when the parser encounters an unexpected character.

`#include <ini_configuration.hh>`

Inheritance diagram for ini::UnexpectedCharacter:



Public Member Functions

- UnexpectedCharacter (const std::istream::int_type character_init, const std::istream::pos_type position_init) throw ()

  Constructs a new UnexpectedCharacter exception.
- UnexpectedCharacter (const UnexpectedCharacter &original) throw ()

  Constructs a new UnexpectedCharacter instance by copying another one.
- virtual ∼UnexpectedCharacter () throw ()

  Destructs a .
- UnexpectedCharacter & operator= (const UnexpectedCharacter &original) throw ()

  Copies an UnexpectedCharacter.
- virtual const char ∗ what () const throw ()

  Returns a description of the error hat occurred.

Additional Inherited Members

5.9.1 Detailed Description

The exception that is thrown when the parser encounters an unexpected character.

Definition at line 81 of file ini_configuration.hh.

5.9.2 Constructor & Destructor Documentation

5.9.2.1 ini::UnexpectedCharacter::UnexpectedCharacter (const std::istream::int_type *character_init*, const std::istream::pos_type *position_init*) throw ()

Constructs a new UnexpectedCharacter exception.

Parameters

character_init	The unexpected character.
position_init	The position of the character in the input stream.

Definition at line 59 of file ini_configuration.cc.

5.9.2.2 ini::UnexpectedCharacter::UnexpectedCharacter (const UnexpectedCharacter & *original*) throw ()

Constructs a new UnexpectedCharacter instance by copying another one.

Parameters

original	The instance that is copied.

Definition at line 96 of file ini_configuration.cc.

5.9.3 Member Function Documentation

5.9.3.1 UnexpectedCharacter & ini::UnexpectedCharacter::operator= (const UnexpectedCharacter & *original*) throw ()

Copies an UnexpectedCharacter.

Parameters

original	The instance that is copied.

Returns

A reference to this instance.

Definition at line 110 of file ini_configuration.cc.

5.9.3.2 const char ∗ ini::UnexpectedCharacter::what ( ) const throw () `[virtual]`

Returns a description of the error hat occurred.

Returns

A description of the error hat occurred.

Implements ini::ParseException.

Definition at line 121 of file ini_configuration.cc.

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

- /Users/bartsas/Courses/Graphics/SVN/code/cxx/ini_configuration/ini_configuration.hh
- /Users/bartsas/Courses/Graphics/SVN/code/cxx/ini_configuration/ini_configuration.cc

5.10 ini::Value Class Reference

Public Member Functions

- virtual bool **exists** () const =0
- virtual bool **as_int_if_exists** (const std::string &section_name, const std::string &entry_name, int &ret_val) const =0
- virtual bool **as_double_if_exists** (const std::string &section_name, const std::string &entry_name, double &ret_val) const =0
- virtual bool **as_string_if_exists** (const std::string &section_name, const std::string &entry_name, std::string &ret_val) const =0
- virtual bool **as_bool_if_exists** (const std::string &section_name, const std::string &entry_name, bool &ret_-val) const =0
- virtual bool **as_int_tuple_if_exists** (const std::string &section_name, const std::string &entry_name, IntTuple &ret_val) const =0
- virtual bool **as_double_tuple_if_exists** (const std::string &section_name, const std::string &entry_name, DoubleTuple &ret_val) const =0
- virtual void **print** (std::ostream &output_stream) const =0

5.10.1 Detailed Description

Definition at line 305 of file ini_configuration.cc.

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

- /Users/bartsas/Courses/Graphics/SVN/code/cxx/ini_configuration/ini_configuration.cc