BSON C++ API

Version 2.5.1

by Sans Pareil Technologies, Inc.

Generated by Doxygen 1.8.3

Thu Feb 28 2013 10:32:39

Contents

Chapter 1

BSON C++ API Documentation

1.1 Contents

1.2 Overview

This library provides a C++ API for representing BSON data. The code is distributed under the Apache Licence, Version 2. This library has been extracted from a larger cross-platform (mobile and desktop) application library (**UMA**) developed by Sans Pareil Technologies, Inc..

1.3 API

BSON - short for Binary JSON - is the native document format developed by MongoDB. SPT has standardised on using BSON as the data interchange format, thus avoiding the necessity to transform data from an intermediary format to a native format when processing data.

The C++ API provided by MongoDB is integrated a bit too closely with their own server implementation, and is quite inconvenient to work with in a read-write environment as required by client applications. The BSON API presents a familiar map style interface that represents the schemaless design of BSON documents, while allowing the same easy manipulation of data allowed by map implementations. The map data may at any time to converted to its BSON representation and serialised.

1.4 Libraries

UMA uses a few very popular libraries in its implementation. The major libraries used include:

- Poco C++ Libraries (Boost Licence).
- C++ Template Unit Test Framework (TUT - BSD licence). The header files for the unit test framework are included along with the project sources, hence no separate download or install is required to run the Unit Tests.
- Qt - required for qmake only. The Qt API itself is not used in UMA, although it is an excellent API. We use Poco to provide most of the features that QtCore does, and Marmalade provides the cross-platform UI features.

1.4.1 Note

The current version of the library includes the sources from the Poco::Util namespace that deal with JSON parsing. These sources will be part of Poco version 1.5 (no stable release yet). Once Poco 1.5 is released, these sources will be removed from the project.

Chapter 2

Namespace Index

2.1 Namespace List

Here is a list of all namespaces with brief descriptions:

Chapter 3

Hierarchical Index

3.1 Class Hierarchy

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

Chapter 4

Class Index

4.1 Class List

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

Chapter 5

File Index

5.1 File List

Here is a list of all files with brief descriptions:

Chapter 6

Namespace Documentation

6.1 Poco Namespace Reference

Namespaces

- namespace Util

Classes

- class DynamicAnyHolderImpl< Util::JSONArray::Ptr >
- class DynamicAnyHolderImpl< Util::JSONObject::Ptr >

6.2 Poco::Util Namespace Reference

Classes

- class JSONArray
- class JSONConfiguration

  This configuration class extracts configuration properties from a JSON object. An XPath-like syntax for property names is supported to allow full access to the JSON object.
- class JSONObject

  Represents a JSON object.
- class JSONParser

  A class for passing JSON strings or streams.
- class JSONQuery

  Class that can be used to search for a value in a JSON object or array.
- class JSONStringifier

  Helper class for creating a String from a JSON object or array.
- class JSONTemplate

  JSONTemplate is a template engine which uses JSON as input for generating output. There are commands for looping over JSON arrays, include other templates, conditional output, ...

6.3 uma Namespace Reference

Namespaces

- namespace bson

Classes that present a DOM style view of a BSON document.

6.4 uma::bson Namespace Reference

Classes that present a DOM style view of a BSON document.

Namespaces

- namespace io

  Classes that provide IO support for BSON data.

Classes

- class Array

  A class that represents a BSON array type document.
- class BinaryData

  A POD that represents a BSON element of type binary data.
- class Boolean

  A POD that represents a boolean type BSON element value.
- class Code

  A POD that represents a code type BSON element value.
- class CodeWithScope

  A POD that represents a code with document scope BSON element value.
- class DatabaseReference

  A POD that represents a database reference type BSON element value. ∗.
- class Date

  A POD that represents a BSON element value of type date.
- class Document

  A class that represents a BSON Object.
- class Double

  A POD that represents a double value in a BSON element.
- class Element

  A class that represents a BSON element. Stores the name-value mapping for a BSON element in a PIMPL. This enables efficient copy-by-value semantics at the cost of shared data across the various copies of the element.
- class EOO

  A POD that represents a EOO value.
- class Integer

  A POD that represents an integer type BSON element value.
- class Long

  A POD that represents a long (64 bit integer) type BSON element value.
- class Null

  A POD that represents a null BSON element value.
- class Object

  Abstract base class that represents a BSON Object type.
- class ObjectId

  A POD that represents a MongoDB OID type.
- class ODMObject

  Abstract class that presents a more friendly ODM interface than uma::bson::Object.
- class PrimitiveValue

A base value class used to represent primitive type values.

- class RegularExpression

  A POD that represents a regular expression type BSON element value.
- class String

  A POD that represents a string value in a BSON element.
- class Symbol

  A POD that represents a programming language symbol type BSON element value.
- class Text

  A class that holds a string. BSON string/text values can be large, hence take care when copy constructing or assigning text instances.
- class Timestamp

  A POD that represents a database timestamp BSON element value.
- class Undefined
- class Value

Functions

- UMA_BSON_API bool operator== (const Array &lhs, const Array &rhs)

  Compare two arrays for equality. Arrays are considered equal if they have the same size and the same elements at the same indices in the array.
- bool operator!= (const Array &lhs, const Array &rhs)

  Just the reverse of operator ==.
- UMA_BSON_API std::ofstream & operator<< (std::ofstream &os, const BinaryData &element)

  Writes the contents of the binary data to the specified file output stream.
- std::ostream & operator<< (std::ostream &os, const BinaryData::DataType &type)

  Writes the contents of the binary data to the specified output stream.
- bool operator== (const BinaryData &lhs, const BinaryData &rhs)

  Compare two binary data instances for equality. Compares the data stored in each instance and their types for equality.
- bool operator!= (const BinaryData &lhs, const BinaryData &rhs)

  Just the reverse of operator ==.
- bool operator== (const CodeWithScope &lhs, const CodeWithScope &rhs)

  Compare two code with scope instances for equality. Compares the code and scope values of each instance.
- bool operator!= (const CodeWithScope &lhs, const CodeWithScope &rhs)

  Just the reverse of operator ==.
- bool operator== (const DatabaseReference &lhs, const DatabaseReference &rhs)

  Compare two database references for equality. Compares the names of the collections and the object id values.
- bool operator!= (const DatabaseReference &lhs, const DatabaseReference &rhs)

  Just the reverse of operator ==.
- bool operator< (const Date &lhs, const Date &rhs)

  Compare the two data values for `less-than` operator.
- bool operator> (const Date &lhs, const Date &rhs)

  Compare the two data values for `greater-than` operator.
- bool operator== (const Date &lhs, const Date &rhs)

  Compare two date values for equality.
- bool operator!= (const Date &lhs, const Date &rhs)

  Just the reverse of operator ==.
- UMA_BSON_API bool operator== (const Document &lhs, const Document &rhs)

  Compare two documents for equality.
- bool operator!= (const Document &lhs, const Document &rhs)

  Just the reverse of operator ==.

- template<>
  UMA_BSON_API std::string Element::getSimple< std::string > () const

  Specialised method for getting the value of uma::bson::String.
- template<>
  UMA_BSON_API Element & Element::setValue< std::string > (const std::string &v)

  Specialised method for setting the value to uma::bson::String.
- bool operator< (const Element &lhs, const Element &rhs)

  Comparison operator for ordering element instances.
- UMA_BSON_API bool operator== (const Element &lhs, const Element &rhs)

  Compare two elements for equality. Compares the names and held values of the elements.
- bool operator!= (const Element &lhs, const Element &rhs)

  Just the reverse of operator ==.
- bool operator== (const EOO &, const EOO &)

  Compare two EOO values for equality. Always returns `true`.
- bool operator!= (const EOO &lhs, const EOO &rhs)

  Just the reverse of operator ==.
- bool operator== (const Null &, const Null &)

  Compare two null values for equality. Always returns `true`.
- bool operator!= (const Null &lhs, const Null &rhs)

  Just the reverse of operator ==.
- UMA_BSON_API bool operator== (const Object &lhs, const Object &rhs)

  Compare two objects for equality.
- bool operator!= (const Object &lhs, const Object &rhs)

  Just the reverse of operator ==.
- bool operator< (const ObjectId &lhs, const ObjectId &rhs)

  Less than operator for comparing ObjectId instances.
- bool operator> (const ObjectId &lhs, const ObjectId &rhs)

  Greater than operator for comparing ObjectId instances.
- UMA_BSON_API bool operator== (const ObjectId &lhs, const ObjectId &rhs)

  Compare two object id values for equality. Compares the data in each instance.
- bool operator!= (const ObjectId &lhs, const ObjectId &rhs)

  Just the reverse of operator ==.
- template<typename T >
  bool operator== (const PrimitiveValue< T > &lhs, const PrimitiveValue< T > &rhs)

  Compare the encapsulated values in the specified instances for equality;.
- template<typename T >
  bool operator== (const PrimitiveValue< T > &lhs, const T rhs)

  Compare the encapsulated value in the specified instance for equality with the specified primitive value.
- template<typename T >
  bool operator!= (const PrimitiveValue< T > &lhs, const PrimitiveValue< T > &rhs)

  Just the reverse of operator ==.
- template<typename T >
  bool operator!= (const PrimitiveValue< T > &lhs, const T rhs)

  Just the reverse of operator ==.
- template<typename T >
  bool operator< (const PrimitiveValue< T > &lhs, const PrimitiveValue< T > &rhs)

  Check to see if the value encapsulated in the left hand instance is less than the value in the right hand instance.
- template<typename T >
  bool operator< (const PrimitiveValue< T > &lhs, const T rhs)

  Compare the encapsulated value in the left hand instance with the primitive value specified.
- template<typename T >
  bool operator> (const PrimitiveValue< T > &lhs, const PrimitiveValue< T > &rhs)

Compare the encapsulated values of the two instances.

- template<typename T >
  bool operator> (const PrimitiveValue< T > &lhs, const T rhs)

  Compare the encapsulated value in the left hand instance with the primitive value in the right hand instance.
- bool operator== (const RegularExpression &lhs, const RegularExpression &rhs)

  Compare two regular expressions for equality. Compares the `regex` and `flags` values for equality.
- bool operator!= (const RegularExpression &lhs, const RegularExpression &rhs)

  Just the reverse of operator ==.
- bool operator== (const Text &lhs, const Text &rhs)

  Compare the two text instances for equality.
- bool operator== (const Text &text, const std::string &str)

  Compare the held value with the specified value.
- bool operator< (const Text &lhs, const Text &rhs)

  Compare the two text instances. Compares the held values of both instances.
- bool operator< (const Text &text, const std::string &str)

  Compares the value held in the specified text with the specified string value.
- bool operator!= (const Text &lhs, const Text &rhs)

  Just the reverse of operator ==.
- bool operator== (const Timestamp &lhs, const Timestamp &rhs)

  Compare two timestamps for equality. Compares the timestamp and increment values of both instances for equality.
- bool operator!= (const Timestamp &lhs, const Timestamp &rhs)

  Just the reverse of operator ==.
- bool operator== (const Undefined &, const Undefined &)

  Compare two undefined values for equality. Always returns true.
- bool operator!= (const Undefined &lhs, const Undefined &rhs)

  Just the reverse of operator ==.
- std::ostream & operator<< (std::ostream &os, const Value::Type &type)

  Write integer representation of the value type to the specified output stream. This is primarily intended for use while debugging/logging. Note that this cannot be used to write to BSON, since types are written as single byte char values.
- UMA_BSON_API bool operator== (const Value &lhs, const Value &rhs)

  Compare two values for equality. Default implementation performs a true comparison of the standard value type implementations provided by this API.
- bool operator!= (const Value &lhs, const Value &rhs)

  Just the reverse of operator ==.

6.4.1 Detailed Description

Classes that present a DOM style view of a BSON document. Namespace with classes that present a DOM style view of a BSON document. Unlike the MongoDB C++ API, a uma::bson::Document and uma::bson::Array are not related. A Document presents a `std::map` style interface while an array presents a `std::vector` type interface.

All the value objects used to encapsulate BSON data types use shared pointers to store its data where appropriate, making them easy to pass by value. This also makes them harder to clone. Callers must keep this in mind and treat shared memory with the usual care.

6.4.2 Function Documentation

6.4.2.1 template<> UMA_BSON_API std::string uma::bson::Element::getSimple< std::string > ( ) const

Specialised method for getting the value of uma::bson::String.

6.4.2.2 template<> UMA_BSON_API Element& uma::bson::Element::setValue< std::string > (const std::string & *v*)

Specialised method for setting the value to uma::bson::String.

6.4.2.3 bool uma::bson::operator!= (const Null & *lhs*, const Null & *rhs*) [inline]

Just the reverse of operator ==.

6.4.2.4 bool uma::bson::operator!= (const EOO & *lhs*, const EOO & *rhs*) [inline]

Just the reverse of operator ==.

6.4.2.5 bool uma::bson::operator!= (const Undefined & *lhs*, const Undefined & *rhs*) [inline]

Just the reverse of operator ==.

6.4.2.6 bool uma::bson::operator!= (const Value & *lhs*, const Value & *rhs*) [inline]

Just the reverse of operator ==.

6.4.2.7 template<typename T > bool uma::bson::operator!= (const PrimitiveValue< T > & *lhs*, const PrimitiveValue< T > & *rhs*) [inline]

Just the reverse of operator ==.

6.4.2.8 bool uma::bson::operator!= (const CodeWithScope & *lhs*, const CodeWithScope & *rhs*) [inline]

Just the reverse of operator ==.

6.4.2.9 template<typename T > bool uma::bson::operator!= (const PrimitiveValue< T > & *lhs*, const T *rhs*) [inline]

Just the reverse of operator ==.

6.4.2.10 bool uma::bson::operator!= (const DatabaseReference & *lhs*, const DatabaseReference & *rhs*) [inline]

Just the reverse of operator ==.

6.4.2.11 bool uma::bson::operator!= (const RegularExpression & *lhs*, const RegularExpression & *rhs*) [inline]

Just the reverse of operator ==.

6.4.2.12 bool uma::bson::operator!= (const Date & *lhs*, const Date & *rhs*) [inline]

Just the reverse of operator ==.

6.4.2.13 bool uma::bson::operator!= (const Timestamp & *lhs*, const Timestamp & *rhs*) [inline]

Just the reverse of operator ==.

6.4.2.14 bool uma::bson::operator!= (const Text & *lhs*, const Text & *rhs*) [inline]

Just the reverse of operator ==.

6.4.2.15 bool uma::bson::operator!= (const ObjectId & *lhs*, const ObjectId & *rhs*) [inline]

Just the reverse of operator ==.

6.4.2.16 bool uma::bson::operator!= (const Object & *lhs*, const Object & *rhs*) [inline]

Just the reverse of operator ==.

6.4.2.17 bool uma::bson::operator!= (const BinaryData & *lhs*, const BinaryData & *rhs*) [inline]

Just the reverse of operator ==.

6.4.2.18 bool uma::bson::operator!= (const Array & *lhs*, const Array & *rhs*) [inline]

Just the reverse of operator ==.

6.4.2.19 bool uma::bson::operator!= (const Element & *lhs*, const Element & *rhs*) [inline]

Just the reverse of operator ==.

6.4.2.20 bool uma::bson::operator!= (const Document & *lhs*, const Document & *rhs*) [inline]

Just the reverse of operator ==.

6.4.2.21 bool uma::bson::operator< (const Date & *lhs*, const Date & *rhs*) [inline]

Compare the two data values for `less-than` operator.

Parameters

lhs	The left hand date to compare against
rhs	The right hand date to compare against

Returns

Return `true` if the datetime in the left hand instance is lower than the datetime in the right hand instance.

6.4.2.22 bool uma::bson::operator< (const Text & *lhs*, const Text & *rhs*) [inline]

Compare the two text instances. Compares the held values of both instances.

Parameters

lhs	The left hand text instance to compare
rhs	The right hand text instance to compare

Returns

Return `true` if the value held in the left hand instance is lexically less than the value held in the right hand instance.

6.4.2.23 bool uma::bson::operator< (const ObjectId & *lhs*, const ObjectId & *rhs*) [inline]

Less than operator for comparing ObjectId instances.

Returns

Return `true` if the left hand object id is considered less (earlier) than the right hand object id.

6.4.2.24 template<typename T > bool uma::bson::operator< (const PrimitiveValue< T > & *lhs*, const PrimitiveValue< T > & *rhs*) [inline]

Check to see if the value encapsulated in the left hand instance is less than the value in the right hand instance.

Template Parameters

T	The primitive type encapsulated by this class.

Parameters

lhs	The left hand instance to compare.
rhs	The right hand instance to compare.

Returns

Return `true` if the value in the left hand instance is less than the value in the right hand instance.

6.4.2.25 bool uma::bson::operator< (const Text & *text*, const std::string & *str*) [inline]

Compares the value held in the specified text with the specified string value.

Parameters

text	The text value to compare with the string.
str	The string value to compare against

Returns

Return `true` if the value encapsulated in this instance is lexically lower than the value specified.

6.4.2.26 template<typename T > bool uma::bson::operator< (const PrimitiveValue< T > & *lhs*, const T *rhs*) [inline]

Compare the encapsulated value in the left hand instance with the primitive value specified.

Template Parameters

T	The primitive type encapsulated by this class.

Parameters

lhs	The left hand instance to compare.
rhs	The primitive alue to compare

Returns

Return `true` if the value in the left hand instance is less than the right hand value.

6.4.2.27 bool uma::bson::operator< (const Element & *lhs*, const Element & *rhs*) [inline]

Comparison operator for ordering element instances.

Compares the names of the two elements. May be used by hash containers to order the elements by name.

Parameters

lhs	The left hand element to compare
rhs	The right hand element to compare

Returns

Returns `true` if the name of the left hand element is lexically less than the name of the right hand element.

6.4.2.28 std::ostream& uma::bson::operator<< (std::ostream & *os*, const Value::Type & *type*) [inline]

Write integer representation of the value type to the specified output stream. This is primarily intended for use while debugging/logging. Note that this cannot be used to write to BSON, since types are written as single byte char values.

Parameters

os	The output stream to write to
type	The value to write as an integer to the stream

Returns

The output stream for chaining calls.

6.4.2.29 UMA_BSON_API std::ofstream& uma::bson::operator<< (std::ofstream & *os*, const BinaryData & *element*)

Writes the contents of the binary data to the specified file output stream.

Parameters

os	The file output stream to write the binary data to.
element	The binary data instance to serialise to file.

Returns

The output stream for method chaining.

6.4.2.30 std::ostream& uma::bson::operator<< (std::ostream & *os*, const BinaryData::DataType & *type*) [inline]

Writes the contents of the binary data to the specified output stream.

Parameters

os	The output stream to write the binary data to.
element	The binary data instance to serialise to the stream.

Returns

The output stream for method chaining.

6.4.2.31 bool uma::bson::operator== (const Null & , const Null &) [inline]

Compare two null values for equality. Always returns `true`.

Since

2.0

Parameters

lhs	The left hand value to compare
rhs	The right hand value to compare

Returns

Return `true`

6.4.2.32 bool uma::bson::operator== (const EOO & , const EOO &) [inline]

Compare two EOO values for equality. Always returns `true`.

Since

2.0

Parameters

lhs	The left hand value to compare
rhs	The right hand value to compare

Returns

Return `true`

6.4.2.33 bool uma::bson::operator== (const Undefined & , const Undefined &) [inline]

Compare two undefined values for equality. Always returns true.

Since

2.0

Parameters

lhs	The left hand value to compare
rhs	The right hand value to compare

Returns

Return `true`

6.4.2.34 UMA_BSON_API bool uma::bson::operator== (const Value & *lhs*, const Value & *rhs*)

Compare two values for equality. Default implementation performs a true comparison of the standard value type implementations provided by this API.

Since

2.2

Parameters

lhs	The left hand value to compare
rhs	The right hand value to compare

Returns

Return `true` if the two instances are the same.

6.4.2.35 template<typename T > bool uma::bson::operator== (const PrimitiveValue< T > & *lhs*, const PrimitiveValue< T > & *rhs*) [inline]

Compare the encapsulated values in the specified instances for equality;.

Template Parameters

T	The primitive type encapsulated by this class.

Parameters

lhs	The left hand instance to compare.
rhs	The right hand instance to compare.

Returns

Return `true` if the values are equal

6.4.2.36 bool uma::bson::operator== (const Text & *lhs*, const Text & *rhs*) [inline]

Compare the two text instances for equality.

Parameters

lhs	The left hand text instance to compare
rhs	The right hand text instance to compare

Returns

Return `true` if the held values are the same

6.4.2.37 template<typename T > bool uma::bson::operator== (const PrimitiveValue< T > & *lhs*, const T *rhs*) [inline]

Compare the encapsulated value in the specified instance for equality with the specified primitive value.

Template Parameters

T	The primitive type encapsulated by this class.

Parameters

lhs	The primitive instance to compare.
rhs	The primitive value to compare against

Returns

Return `true` if the values are equal

6.4.2.38 bool uma::bson::operator== (const CodeWithScope & *lhs*, const CodeWithScope & *rhs*) [inline]

Compare two code with scope instances for equality. Compares the code and scope values of each instance.

Since

2.0

Parameters

lhs	The left hand value to compare
rhs	The right hand value to compare

Returns

Return `true` if the two instances are equivalent

6.4.2.39 bool uma::bson::operator== (const DatabaseReference & *lhs*, const DatabaseReference & *rhs*) [inline]

Compare two database references for equality. Compares the names of the collections and the object id values.

Since

2.0

Parameters

lhs	The left hand value to compare
rhs	The right hand value to compare

Returns

Return `true` if the two instances are equivalent.

6.4.2.40 bool uma::bson::operator== (const Text & *text,* const std::string & *str*) [inline]

Compare the held value with the specified value.

Parameters

text	The text value to compare with the string.
str	The string value to compare with

Returns

Return `true` if the held value is same as the specified value.

6.4.2.41 bool uma::bson::operator== (const RegularExpression & *lhs,* const RegularExpression & *rhs*) [inline]

Compare two regular expressions for equality. Compares the `regex` and `flags` values for equality.

Since

2.0

Parameters

lhs	The left hand value to compare
rhs	The right hand value to compare

Returns

Return `true` if the two instances are equivalent.

6.4.2.42 bool uma::bson::operator== (const Date & *lhs,* const Date & *rhs*) [inline]

Compare two date values for equality.

Parameters

lhs	The left hand date to compare against
rhs	The right hand date to compare against

Returns

Return `true` if the datetime values represented are equal.

6.4.2.43 bool uma::bson::operator== (const Timestamp & *lhs,* const Timestamp & *rhs*) [inline]

Compare two timestamps for equality. Compares the timestamp and increment values of both instances for equality.

Since

2.0

Parameters

lhs	The left hand value to compare
rhs	The right hand value to compare

Returns

Return `true` if the two instances have the same values.

6.4.2.44 UMA_BSON_API bool uma::bson::operator== (const ObjectId & *lhs*, const ObjectId & *rhs*)

Compare two object id values for equality. Compares the data in each instance.

Since

2.0

Parameters

lhs	The left hand value to compare
rhs	The right hand value to compare

Returns

Return `true` if the two id values are equivalent.

6.4.2.45 UMA_BSON_API bool uma::bson::operator== (const Object & *lhs*, const Object & *rhs*)

Compare two objects for equality.

Ensures that both objects have same number of elements, and compares each element for equality. Objects are considered equal only if they have equivalent elements at the same positions (index) within the object, or which have the same ordering of elements.

Warning

This method performs a true deep comparison and may be quite heavy weight depending upon the data in the documents being compared.

Parameters

lhs	The left hand value to compare
rhs	The right hand value to compare

Returns

Return `true` if the two instances are equal.

6.4.2.46 bool uma::bson::operator== (const BinaryData & *lhs*, const BinaryData & *rhs*) [inline]

Compare two binary data instances for equality. Compares the data stored in each instance and their types for equality.

Warning

Performance of this method will depend upon the size of the binary data held in the instances.

Since

2.0

Parameters

lhs	The left hand value to compare
rhs	The right hand value to compare

Returns

Return `true` if the two instances are equivalent.

6.4.2.47 UMA_BSON_API bool uma::bson::operator== (const Array & *lhs*, const Array & *rhs*)

Compare two arrays for equality. Arrays are considered equal if they have the same size and the same elements at the same indices in the array.

Warning

This method performs a true deep comparison and will not be efficient.

Since

2.0

Parameters

lhs	The left hand value to compare
rhs	The right hand value to compare

Returns

Return `true` if the two instances have equivalent elements.

6.4.2.48 UMA_BSON_API bool uma::bson::operator== (const Element & *lhs*, const Element & *rhs*)

Compare two elements for equality. Compares the names and held values of the elements.

Since

2.0

Parameters

lhs	The left hand value to compare
rhs	The right hand value to compare

Returns

Return `true` if the two elements have the same name and value.

6.4.2.49 UMA_BSON_API bool uma::bson::operator== (const Document & *lhs*, const Document & *rhs*)

Compare two documents for equality.

Ensures that both documents have same number of elements, and compares each element for equality. Documents are considered equal only if they have equivalent elements at the same positions (index) within the document, or which have the same ordering of elements.

Warning

This method performs a true deep comparison and may be quite heavy weight depending upon the data in the documents being compared.

Since

2.0

Parameters

lhs	The left hand value to compare
rhs	The right hand value to compare

Returns

Return `true` if the two instances are equal.

6.4.2.50 bool uma::bson::operator> (const Date & *lhs*, const Date & *rhs*) [inline]

Compare the two data values for `greater-than` operator.

Parameters

lhs	The left hand date to compare against
rhs	The right hand date to compare against

Returns

Return `true` if the left hand datetime instance is greater than the datetime in the right hand instance.

6.4.2.51 bool uma::bson::operator> (const ObjectId & *lhs*, const ObjectId & *rhs*) [inline]

Greater than operator for comparing ObjectId instances.

Returns

Return `true` if the left hand object id is considered greater (newer) than the right hand object id.

6.4.2.52 template<typename T > bool uma::bson::operator> (const PrimitiveValue< T > & *lhs*, const PrimitiveValue< T > & *rhs*) [inline]

Compare the encapsulated values of the two instances.

Template Parameters

T	The primitive type encapsulated by this class.

Parameters

lhs	The left hand instance to compare against
rhs	The right hand instance to compare against

Returns

Return `true` if the left hand instance is greater than the right hand instance.

6.4.2.53 template<typename T > bool uma::bson::operator> (const PrimitiveValue< T > & *lhs*, const T *rhs*) [inline]

Compare the encapsulated value in the left hand instance with the primitive value in the right hand instance.

Template Parameters

T	The primitive type encapsulated by this class.

Parameters

lhs	The left instance to compare
rhs	The primitivate value to compare

Returns

Return `true` if the encapsulated value is greater than the primitive value.

6.5 uma::bson::io Namespace Reference

Classes that provide IO support for BSON data.

Classes

- class ArrayJsonReader

  A parser for transforming a JSON serialised BSON array into an uma::bson::Array object representation.
- class ArrayJsonWriter

  A streaming writer that transforms a uma::bson::Array to a JSON representation.
- class ArrayReader

  A streaming parser that parses a stream of BSON bytes from an input stream and transforms into a uma::bson::Array model.
- class ArrayWriter

  A serialiser that writes the BSON representation of an array to a output stream.
- class DocumentJsonReader

  A parser that transforms a JSON stream into a uma::bson::Document.
- class DocumentJsonWriter

  A streaming writer that transforms a uma::bson::Document to a JSON representation.
- class DocumentReader

  A streaming parser that parses a stream of BSON bytes from an input stream and transforms into a uma::bson::-Document model.
- class DocumentWriter

  A serialiser that writes the BSON representation of a document to an output stream.
- class JsonReader

A base streaming parser that parses a stream of JSON bytes from an input stream and transforms into a object model.

- class JsonWriter

  A streaming writer that transforms an object model into a JSON representation.

- class ObjectReader
- class ObjectWriter
- class Reader

  A base streaming parser that parses a stream of BSON bytes from an input stream and transforms into a object model.

- class Writer

6.5.1 Detailed Description

Classes that provide IO support for BSON data. Namespace with classes that provide IO support for BSON data. Provides support for reading from BSON data, generating BSON data, as well as JSON IO support. The uma::bson::Document and uma::bson::Array classes provide support for BSON/JSON IO using classes in this namespace. Clients would normally not need to directly interact with these classes.

Chapter 7

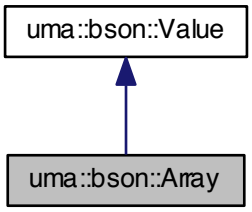
Class Documentation

7.1 uma::bson::Array Class Reference

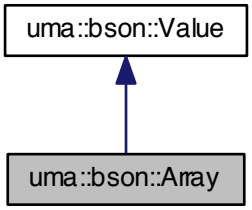
A class that represents a BSON array type document.

`#include <Array.h>`

Inheritance diagram for uma::bson::Array:



Collaboration diagram for uma::bson::Array:



Public Types

- typedef std::vector< Element >
  ::const_iterator ConstantIterator

  A constant iterator for the elements in the array.

Public Member Functions

- Array ()

  Default constructor. Unlike Document no object id is assigned.
- bool isEmpty () const

  Check to see if the array is empty (has no elements)
- size_t size () const

  Return the total number of fields->(top-level) in this array.
- ConstantIterator begin () const

  Return a constant iterator to the first element in the array.
- ConstantIterator end () const

  Return a constant iterator to the last element in the array.
- const Element & at (const size_t index) const

  Return the element at the specified index in the array as a constant reference.
- Element & at (const size_t index)

  Return the element at the specified index in the array as a reference.
- Element & operator[] (const size_t index)

  Return a reference to the element at the specified index in the array.
- const Element & operator[] (const size_t index) const

  Alias for const.
- Array & add (const Element &element, const int index=-1)

  Adds the specified element to the array.
- template<typename DataType >
  Array & add (const DataType &value)

  Add a new element with the specified value to the array.
- template<typename DataType >
  Array & operator<< (const DataType &value)

  Alias for add method.
- Element remove (const int index)

  Remove the element at the specified index.
- void toBson (std::ostream &os) const

  Write a BSON representation of the data in this array to the output stream.
- void toJson (std::ostream &os, bool prettyPrint=false) const

  Generate a JSON representation of the data in the array.
- Array clone () const

  Clone the contents of this array.
- int32_t getSize () const

  Return the size of the bson data held in the array.
- Value::Type getType () const

  Returns the datatype for an array as specified in the BSON specification.

Static Public Member Functions

- static Array fromFile (const std::string &filePath)

  Create a new array instance with the contents of the BSON file at specified path.
- static Array fromBytes (const char *bytes, const int32_t length)

  Create a new array instance with the contents of the `char` array. Can be used to create array instances from MongoDB C/C++ driver functions.
- static Array fromStream (std::istream &is)

  Create a new array instance with the contents read from the specified stream. Usually used to read the BSON data over a network.
- static Array fromJson (std::istream &is)

  Create a new array instance with the JSON data from the specified stream.

Additional Inherited Members

7.1.1 Detailed Description

A class that represents a BSON array type document.

An array stores its elements using sequential numbers as the element/field names. All data is stored as a shared pointer to a PIMPL class to allow efficient pass by value semantics. To properly clone/copy an array use the clone method.

Date

Created 2012/09/06 18:41

Copyright

Author

Rakesh

Version

Id:

Array.h 203 2013-02-15 12:23:15Z spt

7.1.2 Member Typedef Documentation

7.1.2.1 typedef std::vector<Element>::const_iterator uma::bson::Array::ConstantIterator

A constant iterator for the elements in the array.

7.1.3 Constructor & Destructor Documentation

7.1.3.1 uma::bson::Array::Array ( ) [inline]

Default constructor. Unlike Document no object id is assigned.

7.1.4 Member Function Documentation

7.1.4.1 Array& uma::bson::Array::add (const Element & *element*, const int *index* = -1) [inline]

Adds the specified element to the array.

Note that any name specified to the element is ignored. Moreover, the name of the element is modified to indciate its index in the array.

Parameters

element	The element to add to the array
index	The index at which to add the element. This enables user defined ordering of the array contents.

Returns

The current array instance to enable method chaining.

7.1.4.2 template<typename DataType > Array& uma::bson::Array::add (const DataType & *value*) [inline]

Add a new element with the specified value to the array.

Template Parameters

DataType	The type of value to be added to the array.

Parameters

value	The new value to add to the array

Returns

The current array instance to enable method chaining.

7.1.4.3 const Element& uma::bson::Array::at (const size_t *index*) const [inline]

Return the element at the specified index in the array as a constant reference.

Parameters

index	The array index from which to retrieve the element.

Returns

The element at the specified index.

Exceptions

std::out_of_range	If the specified index is invalid.

7.1.4.4 Element& uma::bson::Array::at (const size_t *index*) [inline]

Return the element at the specified index in the array as a reference.

Parameters

index	The array index from which to retrieve the element.

Returns

The element at the specified index

Exceptions

std::out_of_range	If the specified index is invalid.

7.1.4.5 ConstantIterator uma::bson::Array::begin ( ) const [inline]

Return a constant iterator to the first element in the array.

Since

2.0

Returns

The iterator to the first element.

7.1.4.6 Array uma::bson::Array::clone ( ) const

Clone the contents of this array.

This is the recommended way to copy an array. The default copy constructor and assignment operator return array instances that share the internal data. This method returns an array instance with the equivalent but different instances of the internal data.

Warning

This method has not been optimised. The current implementation serialises the array to a BSON stream, and then de-serialises the BSON stream into a new array instance.

Since

1.3

Returns

An array instance with all the internal data duplicated.

7.1.4.7 ConstantIterator uma::bson::Array::end ( ) const [inline]

Return a constant iterator to the last element in the array.

Since

2.0

Returns

The iterator to the last element.

7.1.4.8 static Array uma::bson::Array::fromBytes (const char * *bytes*, const int32_t *length*) [static]

Create a new array instance with the contents of the `char` array. Can be used to create array instances from MongoDB C/C++ driver functions.

Parameters

bytes	The char array containing the BSON data
length	The total length of the char array

Returns

The array object representing the BSON data.

7.1.4.9 static Array uma::bson::Array::fromFile (const std::string & *filePath*) [static]

Create a new array instance with the contents of the BSON file at specified path.

Parameters

filePath	The fully qualified name of the file to read.

Returns

The array object representing the BSON data.

7.1.4.10 static Array uma::bson::Array::fromJson (std::istream & *is*) [static]

Create a new array instance with the JSON data from the specified stream.

Parameters

is	The input stream to read the JSON data from

Returns

The array object representing the BSON data.

7.1.4.11 static Array uma::bson::Array::fromStream (std::istream & *is*) [static]

Create a new array instance with the contents read from the specified stream. Usually used to read the BSON data over a network.

Parameters

is	The input stream to read the BSON data from

Returns

The array object representing the BSON data.

7.1.4.12 int32_t uma::bson::Array::getSize ( ) const [inline], [virtual]

Return the size of the bson data held in the array.

Returns

The size of the bson data array

Implements uma::bson::Value.

7.1.4.13 Value::Type uma::bson::Array::getType ( ) const [inline], [virtual]

Returns the datatype for an array as specified in the BSON specification.

Returns

The type for an array.

Implements uma::bson::Value.

7.1.4.14 bool uma::bson::Array::isEmpty ( ) const [inline]

Check to see if the array is empty (has no elements)

Returns

Return `true` if there are no child elements.

7.1.4.15 template<typename DataType > Array& uma::bson::Array::operator<< (const DataType & *value*) [inline]

Alias for add method.

7.1.4.16 Element& uma::bson::Array::operator[] (const size_t *index*)

Return a reference to the element at the specified index in the array.

If no element exists at the specified index, an empty element with value uma::bson::Undefined is created and stored at the specified index. A total of `index` - size() elements will be added to the array.

Parameters

index	The index of the element within the array

Returns

The reference to the existing element or newly created one.

7.1.4.17 const Element& uma::bson::Array::operator[] (const size_t *index*) const [inline]

Alias for const.

7.1.4.18 Element uma::bson::Array::remove (const int *index*) [inline]

Remove the element at the specified index.

Parameters

index	The index of the element to remove from the array.

Returns

A copy of the element that was removed.

Exceptions

Poco::NotFoundException	If no element exists at the specified index.

7.1.4.19 size_t uma::bson::Array::size ( ) const [inline]

Return the total number of fields->(top-level) in this array.

Returns

The total number of fields->

7.1.4.20 void uma::bson::Array::toBson (std::ostream & *os*) const

Write a BSON representation of the data in this array to the output stream.

Parameters

os	The output stream to serialise the BSON data to.

7.1.4.21 void uma::bson::Array::toJson (std::ostream & *os*, bool *prettyPrint* = `false`) const

Generate a JSON representation of the data in the array.

Note that the JSON generated is based closely upon the BSON specification and is not in a "natural" JSON format. The format was chosen to make it easy to re-parse the JSON back into a array.

Parameters

os	The output stream to serialise the JSON data to.
prettyPrint	Indicate whether the output should contain indentation and line breaks for readability.

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

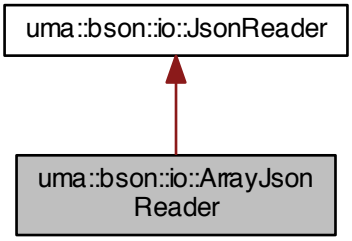
- /Users/rakesh/projects/opensource/mongoviewer/bson/src/api/uma/bson/Array.h

7.2 uma::bson::io::ArrayJsonReader Class Reference

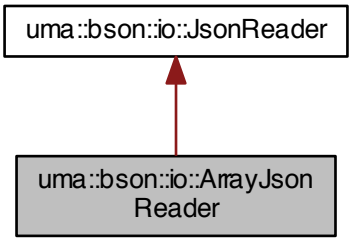
A parser for transforming a JSON serialised BSON array into an uma::bson::Array object representation.

`#include <ArrayJsonReader.h>`

Inheritance diagram for uma::bson::io::ArrayJsonReader:



Collaboration diagram for uma::bson::io::ArrayJsonReader:



Public Member Functions

- Array parse (const std::string &filePath)

  Parses the contents of the file at the path specified into an array instance.
- Array parse (std::istream &fin)

  Reads a JSON representation of a BSON array from the stream.

Additional Inherited Members

7.2.1 Detailed Description

A parser for transforming a JSON serialised BSON array into an uma::bson::Array object representation.

Note that this parser will process only JSON data that was created in the format created by JsonWriter.

Copyright

Date

Created 2012/09/30 06:45

Author

Rakesh

Version

Id:

ArrayJsonReader.h 167 2012-12-13 22:11:47Z spt

7.2.2 Member Function Documentation

7.2.2.1 Array uma::bson::io::ArrayJsonReader::parse (const std::string & *filePath*)

Parses the contents of the file at the path specified into an array instance.

Parameters

filePath	The fully qualified path to the JSON file.

Returns

Document The array instance that represents the JSON data in the file.

7.2.2.2 Array uma::bson::io::ArrayJsonReader::parse (std::istream & *fin*)

Reads a JSON representation of a BSON array from the stream.

Parameters

fin	The input stream to read from.

Returns

Array The parsed array instance.

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

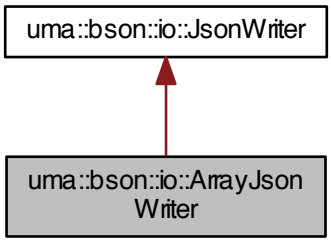
- /Users/rakesh/projects/opensource/mongoviewer/bson/src/api/uma/bson/io/ArrayJsonReader.h

7.3 uma::bson::io::ArrayJsonWriter Class Reference

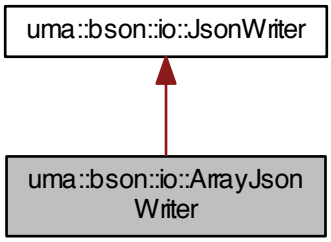
A streaming writer that transforms a uma::bson::Array to a JSON representation.

`#include <ArrayJsonWriter.h>`

Inheritance diagram for uma::bson::io::ArrayJsonWriter:



Collaboration diagram for uma::bson::io::ArrayJsonWriter:



Public Member Functions

- ArrayJsonWriter (const Array &arr, std::ostream &os, bool prettyPrint=false)

  Create a new instance of the writer used to serialise the specified array as JSON data to the specified output stream.
- void write ()

  Serialise the array to the output stream.

Additional Inherited Members

7.3.1 Detailed Description

A streaming writer that transforms a uma::bson::Array to a JSON representation.

Date

Created 2012/09/24 13:18

Copyright

Author

Rakesh

Version

Id:

ArrayJsonWriter.h 167 2012-12-13 22:11:47Z spt

7.3.2 Constructor & Destructor Documentation

7.3.2.1 uma::bson::io::ArrayJsonWriter::ArrayJsonWriter (const Array & *arr*, std::ostream & *os*, bool *prettyPrint* = `false`)

Create a new instance of the writer used to serialise the specified array as JSON data to the specified output stream.

Parameters

arr	The array to serialise as JSON
os	The output stream to serialise to.
prettyPrint	Indicate whether the JSON output should contain indentation and new lines. Defaults to `false` which produces compact (non-pretty) output.

7.3.3 Member Function Documentation

7.3.3.1 void uma::bson::io::ArrayJsonWriter::write ( )

Serialise the array to the output stream.

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

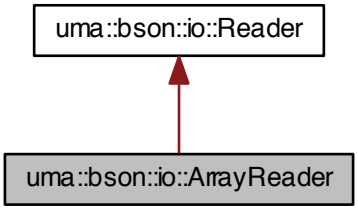
- /Users/rakesh/projects/opensource/mongoviewer/bson/src/api/uma/bson/io/ArrayJsonWriter.h

7.4 uma::bson::io::ArrayReader Class Reference

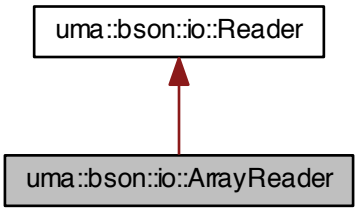
A streaming parser that parses a stream of BSON bytes from an input stream and transforms into a uma::bson::Array model.

`#include <ArrayReader.h>`

Inheritance diagram for uma::bson::io::ArrayReader:



Collaboration diagram for uma::bson::io::ArrayReader:



Public Member Functions

- Array parse (const std::string &filePath)

  Parses the contents of the file at the path specified into a array instance.
- Array parse (std::istream &fin)

  Parses the BSON data bytes from the specified input stream into a array.

Additional Inherited Members

7.4.1 Detailed Description

A streaming parser that parses a stream of BSON bytes from an input stream and transforms into a uma::bson::Array model.

Date

Created 2012/10/02 19:46

Copyright

Author

Rakesh

Version

Id:

ArrayReader.h 167 2012-12-13 22:11:47Z spt

7.4.2 Member Function Documentation

7.4.2.1 Array uma::bson::io::ArrayReader::parse (const std::string & *filePath*)

Parses the contents of the file at the path specified into a array instance.

Parameters

filePath	The fully qualified path to the BSON file.

Returns

Array The array instance that represents the BSON data in the file.

7.4.2.2 Array uma::bson::io::ArrayReader::parse (std::istream & *fin*)

Parses the BSON data bytes from the specified input stream into a array.

Parameters

fin	The input stream (file or network) from which the BSON data is to be read.

Returns

Array The array instance that represents the BSON data from the stream.

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

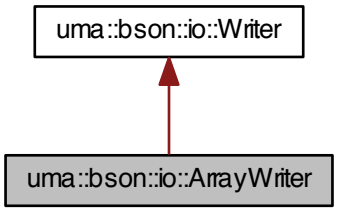
- /Users/rakesh/projects/opensource/mongoviewer/bson/src/api/uma/bson/io/ArrayReader.h

7.5 uma::bson::io::ArrayWriter Class Reference

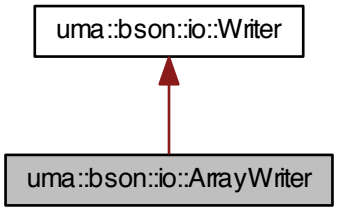
A serialiser that writes the BSON representation of an array to a output stream.

`#include <ArrayWriter.h>`

Inheritance diagram for uma::bson::io::ArrayWriter:



Collaboration diagram for uma::bson::io::ArrayWriter:



Public Member Functions

- ArrayWriter (const Array &arr, std::ostream &os)

  Create a new array writer instance for serialising the specified array instance to the specified output stream.
- void write ()

  Serialise the document to the output stream.

Additional Inherited Members

7.5.1 Detailed Description

A serialiser that writes the BSON representation of an array to a output stream.

Date

Created 2012/10/02 19:12

Copyright

Author

Rakesh

Version

Id:

ArrayWriter.h 167 2012-12-13 22:11:47Z spt

7.5.2 Constructor & Destructor Documentation

7.5.2.1 uma::bson::io::ArrayWriter::ArrayWriter (const Array & *arr*, std::ostream & *os*)

Create a new array writer instance for serialising the specified array instance to the specified output stream.

Parameters

arr	The array to serialise as BSON
os	The stream to serialise the BSON data to

7.5.3 Member Function Documentation

7.5.3.1 void uma::bson::io::ArrayWriter::write ( )

Serialise the document to the output stream.

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

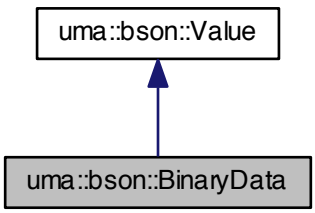
- /Users/rakesh/projects/opensource/mongoviewer/bson/src/api/uma/bson/io/ArrayWriter.h

7.6 uma::bson::BinaryData Class Reference

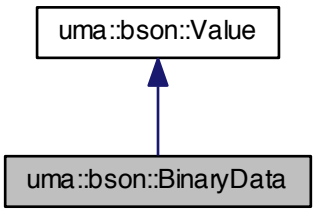
A POD that represents a BSON element of type binary data.

`#include <BinaryData.h>`

Inheritance diagram for uma::bson::BinaryData:



Collaboration diagram for uma::bson::BinaryData:



Public Types

- enum DataType {
  General = 0, Function = 1, ByteArrayDeprecated = 2, UUIDDeprecated = 3,
  UUID = 4, MD5 = 5, Custom = 128 }

  Enumeration of the types used to identify the type of binary content.
- typedef std::vector< char > Buffer

  The container used to store the binary data.

Public Member Functions

- BinaryData (const DataType dt=BinaryData::General)

  Default constructor.
- BinaryData (const char *&bytes, const int length, const DataType dt=BinaryData::General)

  Create a new instance with the data specified.
- BinaryData (std::istream &is, const DataType dt=BinaryData::General)

  Create a new instance from the contents of the specified stream.
- BinaryData (const std::string &filePath, const DataType dt=BinaryData::General)

  Create a new instance with the contents of the file at specified path.
- const Buffer & getData () const

  Return the data stored in this instance as a constant reference.
- Buffer & getData ()

  Return the data stored in this instance as a non-constant reference.
- BinaryData & setData (const Buffer &buffer, const DataType dataType=BinaryData::General)

  Set the binary data for this instance.
- DataType getDataType () const

  Return the data type for the binary data.
- int32_t getSize () const

  Return the size of the bson representation of the binary data.
- Value::Type getType () const

  Return the type that represents binary data as per BSON specifications.

Additional Inherited Members

7.6.1 Detailed Description

A POD that represents a BSON element of type binary data.

This is just an array of bytes (char) stored in a std::vector. The recommended way to fill/populate the BinaryData::-Buffer is to create an empty instance of this class, get a non-constant reference to the buffer and populate it. The usual caveat about reserving space in the vector before filling applies.

Date

Created 2012/09/06 18:50

Copyright

Author

Rakesh

Version

Id:

BinaryData.h 174 2012-12-15 13:13:56Z spt

7.6.2 Member Typedef Documentation

7.6.2.1 typedef std::vector<char> uma::bson::BinaryData::Buffer

The container used to store the binary data.

7.6.3 Member Enumeration Documentation

7.6.3.1 enum uma::bson::BinaryData::DataType

Enumeration of the types used to identify the type of binary content.

Enumerator

General Binary / Generic data to be used by all clients/tools.

Function

ByteArrayDeprecated use General instead

UUIDDeprecated deprecated

UUID language-independent UUID format across all drivers

MD5

Custom User defined type.

7.6.4 Constructor & Destructor Documentation

7.6.4.1 uma::bson::BinaryData::BinaryData (const DataType *dt* = BinaryData::General) [inline]

Default constructor.

Parameters

dt	The binary data type to assign to this instance. Defaults to `DataType::General`

7.6.4.2 uma::bson::BinaryData::BinaryData (const char *& *bytes*, const int *length*, const DataType *dt* = BinaryData::General) [inline]

Create a new instance with the data specified.

Parameters

length	The total length in bytes of the binary content
data	The binary data
dataType	The data type for the binary content

7.6.4.3 uma::bson::BinaryData::BinaryData (std::istream & *is*, const DataType *dt* = BinaryData::General) [inline]

Create a new instance from the contents of the specified stream.

Parameters

is	The input stream from which the binary data is to be read.
dataType	The data type for the binary content

7.6.4.4 uma::bson::BinaryData::BinaryData (const std::string & *filePath*, const DataType *dt* = BinaryData::General) [inline]

Create a new instance with the contents of the file at specified path.

Parameters

filePath	The fully qualified path to the file.

Exceptions

Poco::Exception	If errors are encountered while reading the file.

7.6.5 Member Function Documentation

7.6.5.1 const Buffer& uma::bson::BinaryData::getData ( ) const [inline]

Return the data stored in this instance as a constant reference.

Returns

The binary data.

7.6.5.2 Buffer& uma::bson::BinaryData::getData ( ) [inline]

Return the data stored in this instance as a non-constant reference.

Returns

The binary data.

7.6.5.3 DataType uma::bson::BinaryData::getDataType ( ) const [inline]

Return the data type for the binary data.

Returns

The binary data type

7.6.5.4 int32_t uma::bson::BinaryData::getSize ( ) const [inline], [virtual]

Return the size of the bson representation of the binary data.

The size of the bson representation is 4 bytes for the length of the data, plus 1 byte for the binary data type, plus the size of the buffer itself.

Returns

The size of the bson representation of the vector holding the binary data.

Implements uma::bson::Value.

7.6.5.5 Value::Type uma::bson::BinaryData::getType ( ) const [inline], [virtual]

Return the type that represents binary data as per BSON specifications.

Returns

Returns uma::bson::Value::Type::BinData

Implements uma::bson::Value.

7.6.5.6 BinaryData& uma::bson::BinaryData::setData (const Buffer & *buffer*, const DataType *dataType* = BinaryData::General) [inline]

Set the binary data for this instance.

Warning

Note that this method involves copying the specified data. This can be a very expensive method and should be used with care. Recommended method is to retrieve the non-constant reference to the data and clear/fill it. If populating a fresh instance, default construct an instance, fetch reference to the data and fill it.

Parameters

buffer	The data that will replace the current data.
dataType	The data type for the new binary data.

Returns

The current instance for method chaining.

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

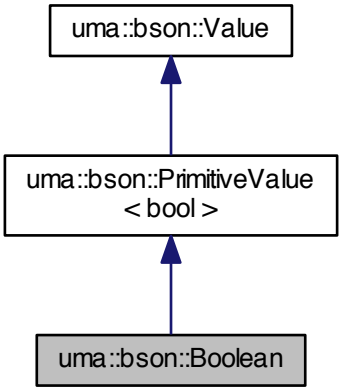
- /Users/rakesh/projects/opensource/mongoviewer/bson/src/api/uma/bson/BinaryData.h

7.7 uma::bson::Boolean Class Reference

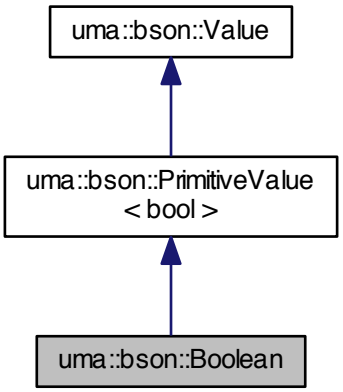
A POD that represents a boolean type BSON element value.

`#include <Boolean.h>`

Inheritance diagram for uma::bson::Boolean:



Collaboration diagram for uma::bson::Boolean:



Public Member Functions

- Boolean ()

  Default CTOR.
- Boolean (const bool v)

  Create a new instance that encapsulates the specified value.
- int32_t getSize () const

  Return the size in bytes occupied by this instance.
- Value::Type getType () const

  Return the type for a boolean value as listed in the BSON specifications.

Additional Inherited Members

7.7.1 Detailed Description

A POD that represents a boolean type BSON element value.

Date

Created 2012/09/13 18:15

Copyright

Author

Rakesh

Version

Id:

Boolean.h 172 2012-12-14 17:21:21Z spt

7.7.2 Constructor & Destructor Documentation

7.7.2.1 uma::bson::Boolean::Boolean ( ) [inline]

Default CTOR.

7.7.2.2 uma::bson::Boolean::Boolean (const bool *v*) [inline], [explicit]

Create a new instance that encapsulates the specified value.

Parameters

v	The boolean to encapsulate

7.7.3 Member Function Documentation

7.7.3.1 int32_t uma::bson::Boolean::getSize ( ) const [inline], [virtual]

Return the size in bytes occupied by this instance.

This is used to compute the total size of a bson uma::bson::Document or uma::bson::Array

Returns

The size in bytes (1)

Implements uma::bson::Value.

7.7.3.2 Value::Type uma::bson::Boolean::getType ( ) const [inline], [virtual]

Return the type for a boolean value as listed in the BSON specifications.

Returns

Returns uma::bson::Value::Type::Boolean

Implements uma::bson::Value.

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

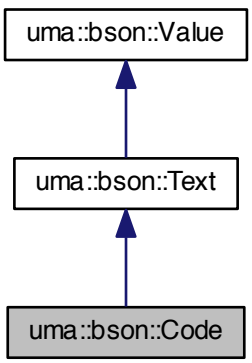
- /Users/rakesh/projects/opensource/mongoviewer/bson/src/api/uma/bson/Boolean.h

7.8 uma::bson::Code Class Reference

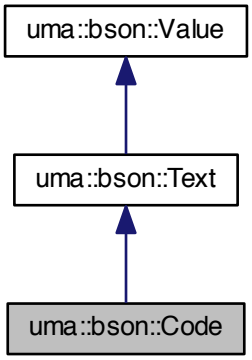
A POD that represents a code type BSON element value.

`#include <Code.h>`

Inheritance diagram for uma::bson::Code:



Collaboration diagram for uma::bson::Code:



Public Member Functions

- Code ()

  Default CTOR.
- Code (const std::string &v)

  Create a new instance with the specified code value.
- Code (const char *data)

  Create a new instance from a C-style char array.
- Value::Type getType () const

  Returns the type for a code instance as listed in the BSON specifications.

Additional Inherited Members

7.8.1 Detailed Description

A POD that represents a code type BSON element value.

Date

Created 2012/09/05 19:55

Copyright

Author

Rakesh

Version

Id:

Code.h 172 2012-12-14 17:21:21Z spt

7.8.2 Constructor & Destructor Documentation

7.8.2.1 uma::bson::Code::Code ( ) [inline]

Default CTOR.

7.8.2.2 uma::bson::Code::Code (const std::string & *v*) [inline]

Create a new instance with the specified code value.

Parameters

v	The code value to encapsulate

7.8.2.3 uma::bson::Code::Code (const char * *data*) [inline]

Create a new instance from a C-style char array.

Parameters

data	The code value to encapsulate

7.8.3 Member Function Documentation

7.8.3.1 Value::Type uma::bson::Code::getType ( ) const [inline], [virtual]

Returns the type for a code instance as listed in the BSON specifications.

Returns

Returns uma::bson::Value::Type::Code

Implements uma::bson::Value.

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

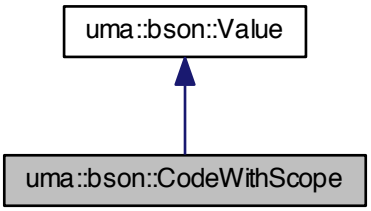
- /Users/rakesh/projects/opensource/mongoviewer/bson/src/api/uma/bson/Code.h

7.9 uma::bson::CodeWithScope Class Reference

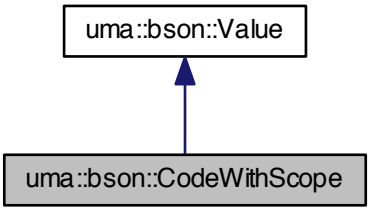
A POD that represents a code with document scope BSON element value.

`#include <CodeWithScope.h>`

Inheritance diagram for uma::bson::CodeWithScope:



Collaboration diagram for uma::bson::CodeWithScope:



Public Member Functions

- CodeWithScope ()

  Default CTOR.
- CodeWithScope (const std::string &c, const Document &doc)

  Create a new instance with the specified code and scope.
- const std::string & getCode () const

  Return the code value stored in this instance as a constant reference.
- std::string & getCode ()

  Return the code value stored in this instance as a non-constant reference.
- const std::string & getValue () const

  Return the code stored in this instance.
- CodeWithScope & setCode (const std::string &v)

  Sets the code encapsulated in this instance.
- const Document & getScope () const

  Returns the scope stored in this instance as a constant reference.
- Document & getScope ()

  Returns the scope stored in this instance as a non-constant reference.
- CodeWithScope & setScope (const Document &doc)

  Sets the scope encapsulated in this instance.

- int32_t getSize () const

  Returns the total size of the BSON representation of this instance.
- Value::Type getType () const

  Returns the type for this instance as listed in the BSON specifications.

Additional Inherited Members

7.9.1 Detailed Description

A POD that represents a code with document scope BSON element value.

Date

Created 2012/09/09 21:52

Copyright

Copyright ©2012, Sans Pareil Technologies, Inc.

Author

Rakesh

Version

Id:

CodeWithScope.h 172 2012-12-14 17:21:21Z spt

7.9.2 Constructor & Destructor Documentation

7.9.2.1 uma::bson::CodeWithScope::CodeWithScope ( ) [inline]

Default CTOR.

7.9.2.2 uma::bson::CodeWithScope::CodeWithScope (const std::string & *c*, const Document & *doc*) [inline]

Create a new instance with the specified code and scope.

Parameters

c	The code to be encapsulated in this instance.
doc	The scope to be encapsulated in this instance.

7.9.3 Member Function Documentation

7.9.3.1 const std::string& uma::bson::CodeWithScope::getCode ( ) const [inline]

Return the code value stored in this instance as a constant reference.

Returns

The code value

7.9.3.2 std::string& uma::bson::CodeWithScope::getCode ( ) [inline]

Return the code value stored in this instance as a non-constant reference.

Returns

The code value

7.9.3.3 const Document& uma::bson::CodeWithScope::getScope ( ) const [inline]

Returns the scope stored in this instance as a constant reference.

Returns

The scope value

7.9.3.4 Document& uma::bson::CodeWithScope::getScope ( ) [inline]

Returns the scope stored in this instance as a non-constant reference.

Returns

The scope value

7.9.3.5 int32_t uma::bson::CodeWithScope::getSize ( ) const [inline], [virtual]

Returns the total size of the BSON representation of this instance.

Returns

The total size if bytes for this instance

Implements uma::bson::Value.

7.9.3.6 Value::Type uma::bson::CodeWithScope::getType ( ) const [inline], [virtual]

Returns the type for this instance as listed in the BSON specifications.

Returns

Returns uma::bson::Value::Type::CodeWScope

Implements uma::bson::Value.

7.9.3.7 const std::string& uma::bson::CodeWithScope::getValue ( ) const [inline]

Return the code stored in this instance.

Returns

The code value

7.9.3.8 CodeWithScope& uma::bson::CodeWithScope::setCode (const std::string & *v*) [inline]

Sets the code encapsulated in this instance.

Parameters

v	The new code value to encapsulate

Returns

The current instance for method chaining

7.9.3.9 CodeWithScope& uma::bson::CodeWithScope::setScope (const Document & *doc*) [inline]

Sets the scope encapsulated in this instance.

Parameters

doc	The new scope value to encapsulate

Returns

This instance for method chaining

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

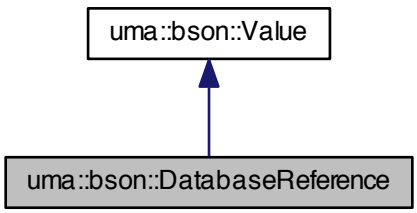
- /Users/rakesh/projects/opensource/mongoviewer/bson/src/api/uma/bson/CodeWithScope.h

7.10 uma::bson::DatabaseReference Class Reference

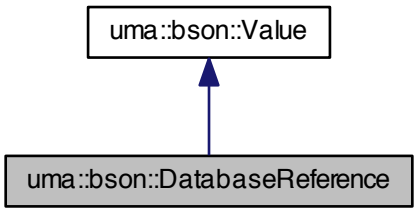
A POD that represents a database reference type BSON element value. ∗.

`#include <DatabaseReference.h>`

Inheritance diagram for uma::bson::DatabaseReference:



Collaboration diagram for uma::bson::DatabaseReference:



Public Member Functions

- DatabaseReference ()

  Default CTOR.
- DatabaseReference (const std::string &col, const ObjectId &id)

  Create a new instance with the specified collection name and OID value.
- const std::string & getCollection () const

  Returns the name of the collection stored in this instance as a constant reference.
- std::string & getCollection ()

  Returns the name of the collection stored in this instance as a non-constant reference.
- DatabaseReference & setCollection (const std::string &col)

  Sets the value of the collection name encapsulated in this instance.
- const ObjectId & getOID () const

  Return the object id of the referenced document as a constant reference.
- ObjectId & getOID ()

  Return the object id of the referenced document as a non-constant reference.
- DatabaseReference & setOID (const ObjectId &id)

  Sets the object id of the referenced document.
- const std::string toString () const

  Returns a string representation of a database reference for logging/debugging purposes.
- int32_t getSize () const

  Returns the size in bytes the BSON representation of this instance occupies.
- Value::Type getType () const

  Returns the type for this instance as listed in the BSON specifications.

Additional Inherited Members

7.10.1 Detailed Description

A POD that represents a database reference type BSON element value. ∗.

Date

Created 2012/09/09 21:44

Copyright

Author

Rakesh

Version

Id:

DatabaseReference.h 172 2012-12-14 17:21:21Z spt

7.10.2 Constructor & Destructor Documentation

7.10.2.1 uma::bson::DatabaseReference::DatabaseReference ( ) [inline]

Default CTOR.

7.10.2.2 uma::bson::DatabaseReference::DatabaseReference (const std::string & *col*, const ObjectId & *id*) [inline]

Create a new instance with the specified collection name and OID value.

Parameters

col	The collection (full namespace for the database and collection) name that the document being referenced resides in.
id	The object id of the referenced document.

7.10.3 Member Function Documentation

7.10.3.1 const std::string& uma::bson::DatabaseReference::getCollection ( ) const [inline]

Returns the name of the collection stored in this instance as a constant reference.

Returns

The collection name.

7.10.3.2 std::string& uma::bson::DatabaseReference::getCollection ( ) [inline]

Returns the name of the collection stored in this instance as a non-constant reference.

Returns

The collection name.

7.10.3.3 const ObjectId& uma::bson::DatabaseReference::getOID ( ) const [inline]

Return the object id of the referenced document as a constant reference.

Returns

The OID value

7.10.3.4 ObjectId& uma::bson::DatabaseReference::getOID ( ) [inline]

Return the object id of the referenced document as a non-constant reference.

Returns

The OID value

7.10.3.5 int32_t uma::bson::DatabaseReference::getSize ( ) const [inline], [virtual]

Returns the size in bytes the BSON representation of this instance occupies.

Returns

The size in bytes of the BSON data

Implements uma::bson::Value.

7.10.3.6 Value::Type uma::bson::DatabaseReference::getType ( ) const [inline], [virtual]

Returns the type for this instance as listed in the BSON specifications.

Returns

Returns uma::bson::Value::Type::DbRef

Implements uma::bson::Value.

7.10.3.7 DatabaseReference& uma::bson::DatabaseReference::setCollection (const std::string & *col*) [inline]

Sets the value of the collection name encapsulated in this instance.

Parameters

col	The new collection name value

Returns

This instance for method chaining.

7.10.3.8 DatabaseReference& uma::bson::DatabaseReference::setOID (const ObjectId & *id*) [inline]

Sets the object id of the referenced document.

Parameters

id	The new object id to encapsulate

Returns

This instance for method chaining.

7.10.3.9 const std::string uma::bson::DatabaseReference::toString ( ) const [inline]

Returns a string representation of a database reference for logging/debugging purposes.

Returns

The textual representation

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

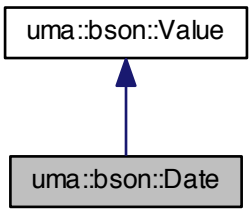
- /Users/rakesh/projects/opensource/mongoviewer/bson/src/api/uma/bson/DatabaseReference.h

7.11 uma::bson::Date Class Reference

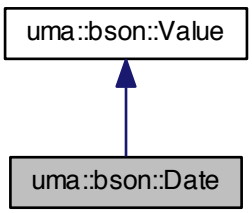
A POD that represents a BSON element value of type date.

`#include <Date.h>`

Inheritance diagram for uma::bson::Date:



Collaboration diagram for uma::bson::Date:



Public Member Functions

- Date ()

  Default CTOR.
- Date (const int32_t time)

  Create a new instance from a UNIX time value.
- Date (const int64_t time)

  Initialise with millisecond precision time since epoch time.
- Date (const Poco::Timestamp &ts)

  Create a new date instance with the specified timestamp.
- const Poco::Timestamp & getValue () const

  Return the time encapsulated in this instance.
- Date & setValue (const Poco::Timestamp &ts)

  Set the time to be encapsulated in this instance.
- const std::string toString () const

  Returns a string representation of the time for logging/debugging purposes.
- int32_t getSize () const

  Returns the size in bytes of the BSON representation of this instance.
- Value::Type getType () const

  Returns the type for this instance as listed in the BSON specifications.

Additional Inherited Members

7.11.1 Detailed Description

A POD that represents a BSON element value of type date.

Dates contain millisecond level precise date-time values. Note that the `Poco::Timestamp` used to represent the date-time is capable of storing date-time with microsecond precision.

Date

Created 2012/09/06 20:13

Copyright

Copyright ©2012, Sans Pareil Technologies, Inc.

Author

Rakesh

Version

Id:

Date.h 172 2012-12-14 17:21:21Z spt

7.11.2 Constructor & Destructor Documentation

7.11.2.1 uma::bson::Date::Date ( ) [inline]

Default CTOR.

7.11.2.2 uma::bson::Date::Date (const int32_t *time*) [inline], [explicit]

Create a new instance from a UNIX time value.

Parameters

time	The seconds since UNIX epoch time.

7.11.2.3 uma::bson::Date::Date (const int64_t *time*) [inline], [explicit]

Initialise with millisecond precision time since epoch time.

Parameters

time	The milliseconds elapsed since UNIX epoch time.

7.11.2.4 uma::bson::Date::Date (const Poco::Timestamp & *ts*) [inline], [explicit]

Create a new date instance with the specified timestamp.

Parameters

ts	The timestamp to initialise this date with.

7.11.3 Member Function Documentation

7.11.3.1 int32_t uma::bson::Date::getSize ( ) const [inline], [virtual]

Returns the size in bytes of the BSON representation of this instance.

Returns

Returns 8 - the length of a int64_t value

Implements uma::bson::Value.

7.11.3.2 Value::Type uma::bson::Date::getType ( ) const [inline], [virtual]

Returns the type for this instance as listed in the BSON specifications.

Returns

Returns uma::bson::Value::Type::Date

Implements uma::bson::Value.

7.11.3.3 const Poco::Timestamp& uma::bson::Date::getValue ( ) const [inline]

Return the time encapsulated in this instance.

Returns

The time value

7.11.3.4 Date& uma::bson::Date::setValue (const Poco::Timestamp & *ts*) [inline]

Set the time to be encapsulated in this instance.

Parameters

ts	The new time value

Returns

This instance for method chaining.

7.11.3.5 const std::string uma::bson::Date::toString ( ) const [inline]

Returns a string representation of the time for logging/debugging purposes.

Returns

Returns the date-time in ISO 8601 format.

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

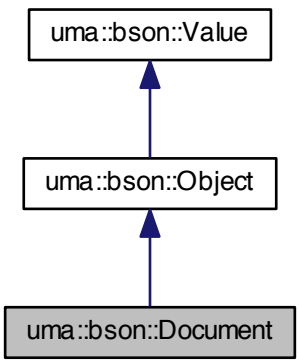
- /Users/rakesh/projects/opensource/mongoviewer/bson/src/api/uma/bson/Date.h

7.12 uma::bson::Document Class Reference

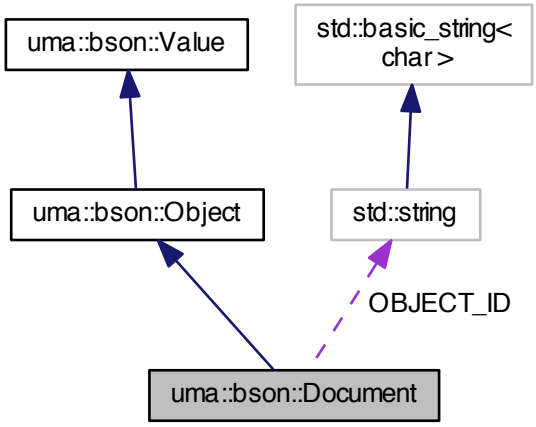
A class that represents a BSON Object.

`#include <Document.h>`

Inheritance diagram for uma::bson::Document:



Collaboration diagram for uma::bson::Document:



Public Types

- typedef std::vector< Element >
  ::const_iterator ConstantIterator

  Constant iterator for iterating over the elements in document.

Public Member Functions

- Document (const ObjectId &oid=ObjectId())

  Default constructor for creating new documents.
- bool isEmpty () const

  Check to see if the document has any elements.
- ConstantIterator begin () const

  Return a constant iterator to the first element in the container that stores the elements in this document.
- ConstantIterator end () const

  Returns a constant iterator to the end of the container that stores the elements in this document.
- const FieldNames getFieldNames () const

  Return a vector of all the element names (top-level only) in this object.
- bool hasElement (const std::string &name) const

  Check to see if a element/field with specified name exists. Note that the check is non-recursive, and hence the name specified must be simple and not dot separated.
- template<typename DataType >
  Element & create (const std::string &name)

  Create an empty element of specified type and return a reference to the newly created element.
- const Element & get (const std::string &name) const

  Return the element with the specified name as a constant reference.
- Element & get (const std::string &name)

  Return the element with the specified name as a reference.
- Element & operator[] (const std::string &name)

  Retrieve the element with specified name.

- const Element & operator[] (const std::string &name) const

  Alias for get(const std::string&).
- Value & getValue (const std::string &name)

  Return the value for the element with the specified name.
- bool hasNestedElement (const std::string &name) const

  Check to see if a element/field with specified name exists in embedded objects/arrays in the document. The nested field name specified must use the dot notation supported by MongoDB.
- const Element & getNestedElement (const std::string &name) const

  Return the nested element from the BSON document.
- Document & set (const Element &value, const int32_t index=-1)

  Add the specified element to this document.
- Document & operator<< (const Element &value)

  Alias for set(uma::bson::Element&, int32_t).
- template<typename DataType >
  Document & set (const std::string &name, const DataType &value, const int32_t index=-1)

  Adds a new element with the specified name and value to this document.
- void setValue (const std::string &name, const Value &value)

  Set the value for the element with the specified name.
- const ObjectId & getObjectId ()

  Return the object identity field for this document.
- Document & setObjectId (const ObjectId &oid=ObjectId())

  Set the `_id` field that holds the object identity for the document.
- Element remove (const std::string &name)

  Remove the element with the specified name from this document.
- size_t size () const

  Return the total number of top-level elements in the document.
- void toBson (std::ostream &os) const

  Serialise the data in this object in BSON format to the specified output stream.
- void toJson (std::ostream &os, bool prettyPrint=false) const

  Generate a JSON representation of the data in the document.
- Document clone () const

  Clone the contents of this document.
- bool isEquivalentTo (const Document &doc) const

  Compare two documents for equivalence.
- int32_t getSize () const

  The size is bytes of the BSON representation of the document.
- template<>
  Document & set (const std::string &name, const double &value, const int32_t index)

  Specialisation of the `set` method for double value.
- template<>
  Document & set (const std::string &name, const std::string &value, const int32_t index)

  Specialisation of the `set` method for a string value.
- template<>
  Document & set (const std::string &name, const bool &value, const int32_t index)

  Specialisation of the `set` method for a boolean value.
- template<>
  Document & set (const std::string &name, const int32_t &value, const int32_t index)

  Specialisation of the `set` method for an integer value.
- template<>
  Document & set (const std::string &name, const int64_t &value, const int32_t index)

  Specialisation of the `set` method for a long value.

Static Public Member Functions

- static Document emptyDocument ()

  Create an empty document without a default assigned OBJECT_ID field.
- static Document fromFile (const std::string &filePath)

  Create a new document instance with the contents of the BSON file at specified path.
- static Document fromBytes (const char *bytes, const int32_t length)

  Create a new document instance with the contents of the `char` array. Can be used to create document instances from MongoDB C/C++ driver functions.
- static Document fromStream (std::istream &is)

  Create a new document instance with the contents read from the specified stream. Usually used to read the BSON data over a network.
- static Document fromJson (std::istream &is)

  Create a new document instance with the JSON data from the specified stream.

Static Public Attributes

- static const std::string OBJECT_ID

  The special ObjectId element/field name as used by MongoDB.

Additional Inherited Members

7.12.1 Detailed Description

A class that represents a BSON Object.

Documents may be top level objects, or sub-objects of a document. All the data in this class is contained in a shared pointer, hence this class may be passed by value without significant overhead. This also makes it easier to work with embedded documents that are retrieved as element values. To properly clone/copy an array use the clone method.

Copyright

Copyright ©2012, Sans Pareil Technologies, Inc.

Date

Created 2012/09/06 18:59

Author

Rakesh

Version

Id:

Document.h 203 2013-02-15 12:23:15Z spt

7.12.2 Member Typedef Documentation

7.12.2.1 typedef std::vector<Element>::const_iterator uma::bson::Document::ConstantIterator

Constant iterator for iterating over the elements in document.

7.12.3 Constructor & Destructor Documentation

7.12.3.1 uma::bson::Document::Document (const ObjectId & *oid* = ObjectId ()) [inline]

Default constructor for creating new documents.

Use to create an new document that will be populated by subsequent invocations of the ∗ set method. The constructor adds a new ObjectId field with the special element name OBJECT_ID.

Parameters

oid	The object id to use for the new document, or a default constructed object id.

7.12.4 Member Function Documentation

7.12.4.1 ConstantIterator uma::bson::Document::begin ( ) const [inline]

Return a constant iterator to the first element in the container that stores the elements in this document.

Since

2.0

Returns

The iterator to the first entry in container.

7.12.4.2 Document uma::bson::Document::clone ( ) const

Clone the contents of this document.

This is the recommended way to copy a document. The default copy constructor and assignment operator return document instances that share the internal data. This method returns a document instance with the equivalent but different instances of the internal data.

Warning

This method has not been optimised. The current implementation serialises the document to a BSON stream, and then de-serialises the BSON stream into a new document instance.

Since

1.3

Returns

A document instance with all the internal data duplicated.

7.12.4.3 template<typename DataType > Element& uma::bson::Document::create (const std::string & *name*) [inline]

Create an empty element of specified type and return a reference to the newly created element.

Use this method to create an element of a specific data type and then populate the value for the newly created element. This will avoid a copy-construction of the value instance when setting an element value.

Warning

Note that if an element exists with the specified name, a reference to the existing element is returned, regardless of the data type held in that element.

Template Parameters

DataType	The type of value to be stored in the element.

Since

Version 2.4

Parameters

name	The name of the element to retrieve.

Returns

The element with the specified name

Exceptions

Poco::NotFoundException	If no element with specified name exists.

7.12.4.4 static Document uma::bson::Document::emptyDocument ( ) [inline], [static]

Create an empty document without a default assigned OBJECT_ID field.

Returns

The new empty document.

7.12.4.5 ConstantIterator uma::bson::Document::end ( ) const [inline]

Returns a constant iterator to the end of the container that stores the elements in this document.

Since

2.0

Returns

The iterator to the end of the container.

7.12.4.6 static Document uma::bson::Document::fromBytes (const char * *bytes*, const int32_t *length*) [static]

Create a new document instance with the contents of the `char` array. Can be used to create document instances from MongoDB C/C++ driver functions.

Parameters

bytes	The char array containing the BSON data
length	The total length of the char array

Returns

The document object representing the BSON data.

7.12.4.7 static Document uma::bson::Document::fromFile (const std::string & *filePath*) [static]

Create a new document instance with the contents of the BSON file at specified path.

Parameters

filePath	The fully qualified name of the file to read.

Returns

The document object representing the BSON data.

7.12.4.8 static Document uma::bson::Document::fromJson (std::istream & *is*) [static]

Create a new document instance with the JSON data from the specified stream.

Parameters

is	The input stream to read the JSON data from

Returns

The document object representing the BSON data.

7.12.4.9 static Document uma::bson::Document::fromStream (std::istream & *is*) [static]

Create a new document instance with the contents read from the specified stream. Usually used to read the BSON data over a network.

Parameters

is	The input stream to read the BSON data from

Returns

The document object representing the BSON data.

7.12.4.10 const Element& uma::bson::Document::get (const std::string & *name*) const [inline]

Return the element with the specified name as a constant reference.

Parameters

name	The name of the element to retrieve.

Returns

The element with the specified name

Exceptions

Poco::NotFoundException	If no element with specified name exists.

7.12.4.11 Element& uma::bson::Document::get (const std::string & *name*) [inline]

Return the element with the specified name as a reference.

Parameters

name	The name of the element to retrieve.

Returns

The element with the specified name

Exceptions

Poco::NotFoundException	If no element with specified name exists.

7.12.4.12 const FieldNames uma::bson::Document::getFieldNames ( ) const [inline], [virtual]

Return a vector of all the element names (top-level only) in this object.

Returns

The vector of element names.

Implements uma::bson::Object.

7.12.4.13 const Element& uma::bson::Document::getNestedElement (const std::string & *name*) const [inline]

Return the nested element from the BSON document.

The element name should be specified using the dot notation supported by MongoDB. Note that array type sub-objects will be traversed by the index number element name.

Parameters

name	The fully qualified (eg. field1.field2.3.field3) name of the embedded element

Returns

The requested embedded element.

Exceptions

Poco::NotFoundException	If the specified nested element is not found.
Poco::Exception	Other Exceptions may be thrown if the name does not properly follow the document structure. For instance if a nested element is an array, and the subsequent element name specified is not of type numeric, and exception will be thrown.

7.12.4.14 const ObjectId& uma::bson::Document::getObjectId ( ) [inline]

Return the object identity field for this document.

Warning

A new object identity will be generated and assigned to this document if one does not yet exist.

Returns

The object id for this document.

7.12.4.15 int32_t uma::bson::Document::getSize ( ) const [inline], [virtual]

The size is bytes of the BSON representation of the document.

Returns

The size of the bson data array

Implements uma::bson::Value.

7.12.4.16 Value& uma::bson::Document::getValue (const std::string & *name*) [inline], [virtual]

Return the value for the element with the specified name.

This method is primarily intended to provide rudimentary ODM (object-document mapping) capabilities in the API. Model objects that extend this class and store fields as Value types may be serialised to BSON seamlessly. See unit test suite for a simple custom model object that is serialised to BSON.

Since

Version 2.2

Parameters

name	The name of the BSON element

Returns

The value for the element.

Exceptions

Poco::NotFoundException	If no element with specified name exists.

Implements uma::bson::Object.

7.12.4.17 bool uma::bson::Document::hasElement (const std::string & *name*) const [inline]

Check to see if a element/field with specified name exists. Note that the check is non-recursive, and hence the name specified must be simple and not dot separated.

Parameters

name	The name of the element/field to check for existence

Returns

Returns `true` if an element exists.

7.12.4.18 bool uma::bson::Document::hasNestedElement (const std::string & *name*) const [inline]

Check to see if a element/field with specified name exists in embedded objects/arrays in the document. The nested field name specified must use the dot notation supported by MongoDB.

Parameters

name	The name of the element/field to check for existence

Returns

Returns `true` if an element exists.

Exceptions

Poco::Exception	Exceptions may be thrown if the name specified does not properly follow the document structure. For instance if a nested element is an array, and the subsequent element name specified is not of type numeric, and exception will be thrown.

7.12.4.19 bool uma::bson::Document::isEmpty ( ) const [inline]

Check to see if the document has any elements.

Returns

Return `true` is there are no elements.

7.12.4.20 bool uma::bson::Document::isEquivalentTo (const Document & *doc*) const

Compare two documents for equivalence.

Documents are considered equivalent if they have the same elements even though the ordering of elements is different. The == operator performs the true equality check, which this method performs equivalence check.

Warning

This method performs a true deep comparison and may be quite heavy weight depending upon the data in the documents being compared.

Since

2.1

Parameters

doc	The document to compare for equivalence.

Returns

Return `true` if the specified document is equivalent to this instance.

7.12.4.21 Document& uma::bson::Document::operator<< (const Element & *value*) [inline]

Alias for set(uma::bson::Element&, int32_t).

7.12.4.22 Element& uma::bson::Document::operator[] (const std::string & *name*)

Retrieve the element with specified name.

Similar to a `std::map` if no element with the specified name exists, a new element with the specified name is added to the document. The element will be assigned a value type `undefined`.

Since

Version 2.2

Parameters

name	The name of the element to retrieve.

Returns

The element with the specified name

7.12.4.23 const Element& uma::bson::Document::operator[] (const std::string & *name*) const [inline]

Alias for get(const std::string&).

7.12.4.24 Element uma::bson::Document::remove (const std::string & *name*) [inline]

Remove the element with the specified name from this document.

Warning

Note that only top-level elements may be removed using this method. Nested elements are not processed.

Parameters

name	The name of the element to remove.

Returns

A copy of the element that was removed.

Exceptions

Poco::NotFoundException	If an element with specified name does not exist in this document.

7.12.4.25 Document& uma::bson::Document::set (const Element & *value,* const int32_t *index* = −1) [inline]

Add the specified element to this document.

Replaces any existing element which has the same name as the specified element.

Warning

If the name of the element is `_id`, it will be inserted at index `0` regardless of the value of `index` specified.

Parameters

value	The new element to add to this document.
index	The optional index at which the new element is to be inserted. This may be used to provide clients control over the order of the elements in this document.

Returns

This instance for method chaining.

7.12.4.26 template<typename DataType > Document& uma::bson::Document::set (const std::string & *name,* const DataType & *value,* const int32_t *index* = −1) [inline]

Adds a new element with the specified name and value to this document.

If an element already exists with the specified name, it is replaced. The optional `index` parameter may be used to control the ordering of the elements in this document.

Template Parameters

DataType	The type of value to be stored in the element.

Parameters

name	The name of the element to be added.
value	The value for the new element.
index	The optional index at which to insert the element.

Returns

This instance for method chaining

7.12.4.27 template<> Document& uma::bson::Document::set (const std::string & *name,* const double & *value,* const int32_t *index*) [inline]

Specialisation of the `set` method for double value.

Parameters

name	The name of the element
value	The double value for the element
index	The optional index to insert the element at.

Returns

This instance for method chaining.

7.12.4.28 template<> Document& uma::bson::Document::set (const std::string & *name*, const std::string & *value*, const int32_t *index*) [inline]

Specialisation of the `set` method for a string value.

Parameters

name	The name of the element
value	The string value for the element
index	The optional index to insert the element at.

Returns

This instance for method chaining.

7.12.4.29 template<> Document& uma::bson::Document::set (const std::string & *name*, const bool & *value*, const int32_t *index*) [inline]

Specialisation of the `set` method for a boolean value.

Parameters

name	The name of the element
value	The boolean value for the element
index	The optional index to insert the element at.

Returns

This instance for method chaining.

7.12.4.30 template<> Document& uma::bson::Document::set (const std::string & *name*, const int32_t & *value*, const int32_t *index*) [inline]

Specialisation of the `set` method for an integer value.

Parameters

name	The name of the element
value	The integer value for the element
index	The optional index to insert the element at.

Returns

This instance for method chaining.

7.12.4.31 template<> Document& uma::bson::Document::set (const std::string & *name*, const int64_t & *value*, const int32_t *index*) [inline]

Specialisation of the `set` method for a long value.

Parameters

name	The name of the element
value	The long value for the element
index	The optional index to insert the element at.

Returns

This instance for method chaining.

7.12.4.32 Document& uma::bson::Document::setObjectId (const ObjectId & *oid* = ObjectId()) [inline]

Set the `_id` field that holds the object identity for the document.

Parameters

oid	The object id for the document.

Returns

This instance for method chaining

7.12.4.33 void uma::bson::Document::setValue (const std::string & *name*, const Value & *value*) [virtual]

Set the value for the element with the specified name.

This method is also used to provide rudimentary ODM capabilities in the API. Model objects that extend this class and store fields as Value types may be deserialised from BSON through this method. See unit test suite for a simple custom model object that is de-serialised from BSON.

Note that from Version 2.3 onwards a default implementation is provided. The default implementation handles setting the value for all the simple types. In particular `Value::Type::Object`, `Value::Type::Array` and consequently `Value::Type::CodeWScope` are not handled and may throw an exception. Despite this clients may still not need to provide an implementation of this method even if they store other object and array fields. The BSON deserialiser implemented in this API retrieves a reference to the current object/array from the destination instance and populates those. Hence if the only code using this method is from the internal BSON deserialiser, there will be no need to provide additional implementation for setting object/array fields.

Since

Version 2.2

Parameters

name	
value	

Exceptions

Poco::InvalidArgument-Exception	Will be thrown if the specified value type does not match the existing value type. May also be throw since this method invokes getValue(const std::string&) at the beginning to fetch the current instance that is to be modified.
Poco::NotImplemented-Exception	If the value specified is a complex type such as `Value::Type::Object` or `Value::Type::Array`.

Reimplemented from uma::bson::Object.

7.12.4.34 size_t uma::bson::Document::size ( ) const [inline]

Return the total number of top-level elements in the document.

Returns

The number of top-level elements.

7.12.4.35 void uma::bson::Document::toBson (std::ostream & *os*) const [virtual]

Serialise the data in this object in BSON format to the specified output stream.

The default implementation uses uma::bson::io::ObjectWriter to serialise the fields returned by getFieldNames to the stream.

Parameters

os	The output stream to serialise the BSON data to.

Reimplemented from uma::bson::Object.

7.12.4.36 void uma::bson::Document::toJson (std::ostream & *os*, bool *prettyPrint* = false) const

Generate a JSON representation of the data in the document.

Note that the JSON generated is based closely upon the BSON specification and is not in a "natural" JSON format. The format was chosen to make it easy to re-parse the JSON back into a document.

Parameters

os	The output stream to serialise the JSON data to.
prettyPrint	Indicate whether the output should contain indentation and line breaks for readability.

7.12.5 Member Data Documentation

7.12.5.1 const std::string uma::bson::Document::OBJECT_ID [static]

The special ObjectId element/field name as used by MongoDB.

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

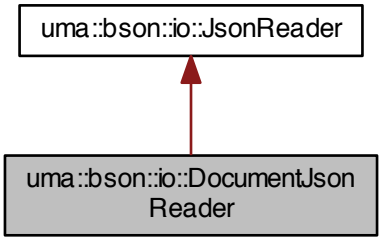
- /Users/rakesh/projects/opensource/mongoviewer/bson/src/api/uma/bson/Document.h

7.13 uma::bson::io::DocumentJsonReader Class Reference

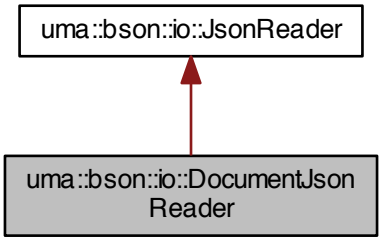
A parser that transforms a JSON stream into a uma::bson::Document.

`#include <DocumentJsonReader.h>`

Inheritance diagram for uma::bson::io::DocumentJsonReader:



Collaboration diagram for uma::bson::io::DocumentJsonReader:



Public Member Functions

- Document parse (const std::string &filePath)

  Parses the contents of the file at the path specified into a document instance.
- Document parse (std::istream &fin)

  Parses the JSON data bytes from the specified input stream into a document.

Additional Inherited Members

7.13.1 Detailed Description

A parser that transforms a JSON stream into a uma::bson::Document.

Note that this parser is designed to work with the JSON output format produced through a DocumentJsonWriter.

Date

Created 2012/09/27 21:46

Copyright

Author

Rakesh

Version

Id:

DocumentJsonReader.h 167 2012-12-13 22:11:47Z spt

7.13.2 Member Function Documentation

7.13.2.1 Document uma::bson::io::DocumentJsonReader::parse (const std::string & *filePath*)

Parses the contents of the file at the path specified into a document instance.

Parameters

filePath	The fully qualified path to the JSON file.

Returns

Document The document instance that represents the JSON data in the file.

7.13.2.2 Document uma::bson::io::DocumentJsonReader::parse (std::istream & *fin*)

Parses the JSON data bytes from the specified input stream into a document.

Parameters

fin	The input stream (file or network) from which the JSON data is to be read.

Returns

Document The document instance that represents the JSON data from the stream.

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

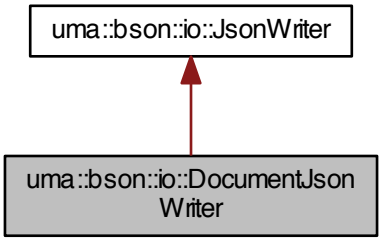
- /Users/rakesh/projects/opensource/mongoviewer/bson/src/api/uma/bson/io/DocumentJsonReader.h

7.14 uma::bson::io::DocumentJsonWriter Class Reference

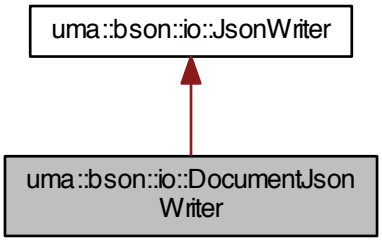
A streaming writer that transforms a uma::bson::Document to a JSON representation.

`#include <DocumentJsonWriter.h>`

Inheritance diagram for uma::bson::io::DocumentJsonWriter:



Collaboration diagram for uma::bson::io::DocumentJsonWriter:



Public Member Functions

- DocumentJsonWriter (const Document &doc, std::ostream &os, bool prettyPrint=false)

  Create a new instance of the writer used to serialise the specified document as JSON data to the specified output stream.
- void write ()

  Serialise the document to the output stream.

Additional Inherited Members

7.14.1 Detailed Description

A streaming writer that transforms a uma::bson::Document to a JSON representation.

Date

Created 2012/09/24 13:18

Copyright

Author

Rakesh

Version

Id:

DocumentJsonWriter.h 167 2012-12-13 22:11:47Z spt

7.14.2 Constructor & Destructor Documentation

7.14.2.1 uma::bson::io::DocumentJsonWriter::DocumentJsonWriter (const Document & *doc,* std::ostream & *os,* bool *prettyPrint* = `false`)

Create a new instance of the writer used to serialise the specified document as JSON data to the specified output stream.

Parameters

doc	The document to serialise as JSON
os	The output stream to serialise to.
prettyPrint	Indicate whether the JSON output should contain indentation and new lines. Defaults to `false` which produces compact (non-pretty) output.

7.14.3 Member Function Documentation

7.14.3.1 void uma::bson::io::DocumentJsonWriter::write ( )

Serialise the document to the output stream.

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

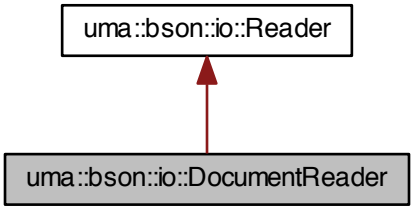
- /Users/rakesh/projects/opensource/mongoviewer/bson/src/api/uma/bson/io/DocumentJsonWriter.h

7.15 uma::bson::io::DocumentReader Class Reference

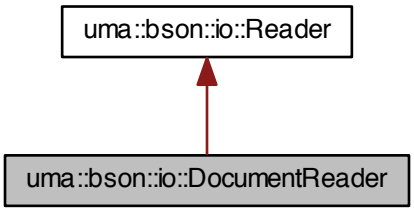
A streaming parser that parses a stream of BSON bytes from an input stream and transforms into a uma::bson::-Document model.

`#include <DocumentReader.h>`

Inheritance diagram for uma::bson::io::DocumentReader:



Collaboration diagram for uma::bson::io::DocumentReader:



Public Member Functions

- Document parse (const std::string &filePath)

  Parses the contents of the file at the path specified into a document instance.
- Document parse (std::istream &fin)

  Parses the BSON data bytes from the specified input stream into a document.

Additional Inherited Members

7.15.1 Detailed Description

A streaming parser that parses a stream of BSON bytes from an input stream and transforms into a uma::bson::-Document model.

Date

Created 2012/09/21 19:49

Copyright

Author

Rakesh

Version

Id:

DocumentReader.h 167 2012-12-13 22:11:47Z spt

7.15.2 Member Function Documentation

7.15.2.1 Document uma::bson::io::DocumentReader::parse (const std::string & *filePath*)

Parses the contents of the file at the path specified into a document instance.

Parameters

filePath	The fully qualified path to the BSON file.

Returns

Document The document instance that represents the BSON data in the file.

7.15.2.2 Document uma::bson::io::DocumentReader::parse (std::istream & *fin*)

Parses the BSON data bytes from the specified input stream into a document.

Parameters

fin	The input stream (file or network) from which the BSON data is to be read.

Returns

Document The document instance that represents the BSON data from the stream.

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

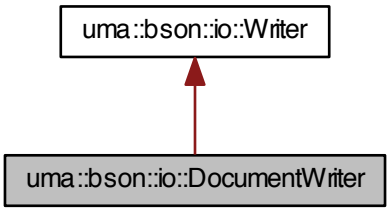
- /Users/rakesh/projects/opensource/mongoviewer/bson/src/api/uma/bson/io/DocumentReader.h

7.16 uma::bson::io::DocumentWriter Class Reference

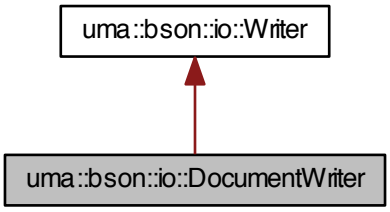
A serialiser that writes the BSON representation of a document to an output stream.

`#include <DocumentWriter.h>`

Inheritance diagram for uma::bson::io::DocumentWriter:



Collaboration diagram for uma::bson::io::DocumentWriter:



Public Member Functions

- DocumentWriter (const Document &doc, std::ostream &os)

  Create a new document writer instance for serialising the specified document instance to the specified output stream.
- void write ()

  Serialise the document to the output stream.

Additional Inherited Members

7.16.1 Detailed Description

A serialiser that writes the BSON representation of a document to an output stream.

Date

Created 2012/09/22 18:30

Copyright

Author

Rakesh

Version

Id:

DocumentWriter.h 167 2012-12-13 22:11:47Z spt

7.16.2 Constructor & Destructor Documentation

7.16.2.1 uma::bson::io::DocumentWriter::DocumentWriter (const Document & *doc,* std::ostream & *os*)

Create a new document writer instance for serialising the specified document instance to the specified output stream.

Parameters

doc	The document to serialise as BSON
os	The stream to serialise the BSON data to

7.16.3 Member Function Documentation

7.16.3.1 void uma::bson::io::DocumentWriter::write ( )

Serialise the document to the output stream.

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

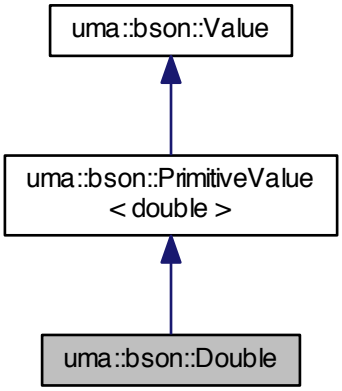
- /Users/rakesh/projects/opensource/mongoviewer/bson/src/api/uma/bson/io/DocumentWriter.h

7.17 uma::bson::Double Class Reference

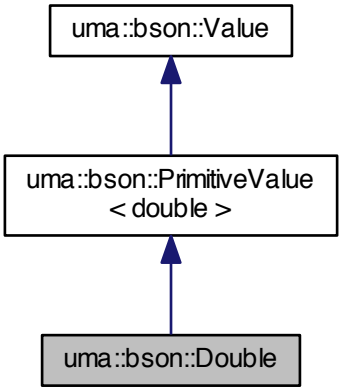
A POD that represents a double value in a BSON element.

`#include <Double.h>`

Inheritance diagram for uma::bson::Double:



Collaboration diagram for uma::bson::Double:



Public Member Functions

- Double ()

  Default CTOR.
- Double (const double v)

  Create a new instance that encapsulates the specified double value.
- int32_t getSize () const

  Returns the number of bytes the BSON representation of this instance.
- Value::Type getType () const

  Returns the type for this instance as listed in the BSON specifications.

Additional Inherited Members

7.17.1 Detailed Description

A POD that represents a double value in a BSON element.

Date

Created 2012/09/13 17:39

Copyright

Author

Rakesh

Version

Id:

Double.h 172 2012-12-14 17:21:21Z spt

7.17.2 Constructor & Destructor Documentation

7.17.2.1 uma::bson::Double::Double ( ) [inline]

Default CTOR.

7.17.2.2 uma::bson::Double::Double (const double *v*) [inline], [explicit]

Create a new instance that encapsulates the specified double value.

Parameters

v	The value to encapsulate in this instance.

7.17.3 Member Function Documentation

7.17.3.1 int32_t uma::bson::Double::getSize ( ) const [inline], [virtual]

Returns the number of bytes the BSON representation of this instance.

Returns

Returns the value from the BSON specification (8).

Implements uma::bson::Value.

7.17.3.2 Value::Type uma::bson::Double::getType ( ) const [inline], [virtual]

Returns the type for this instance as listed in the BSON specifications.

Returns

Returns uma::bson::Value::Type::Double

Implements uma::bson::Value.

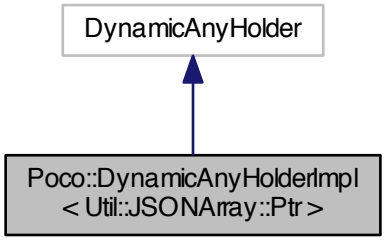
The documentation for this class was generated from the following file:

- /Users/rakesh/projects/opensource/mongoviewer/bson/src/api/uma/bson/Double.h

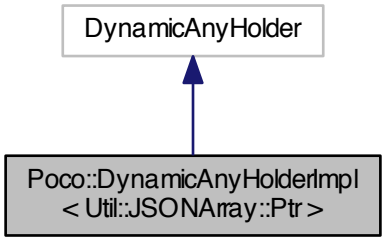
7.18 Poco::DynamicAnyHolderImpl< Util::JSONArray::Ptr > Class Template Reference

`#include <JSONArray.h>`

Inheritance diagram for Poco::DynamicAnyHolderImpl< Util::JSONArray::Ptr >:



Collaboration diagram for Poco::DynamicAnyHolderImpl< Util::JSONArray::Ptr >:



Public Member Functions

- DynamicAnyHolderImpl (const Util::JSONArray::Ptr &val)
- ~DynamicAnyHolderImpl ()
- const std::type_info & type () const
- void convert (Int8 &) const
- void convert (Int16 &) const

- void convert (Int32 &) const
- void convert (Int64 &) const
- void convert (UInt8 &) const
- void convert (UInt16 &) const
- void convert (UInt32 &) const
- void convert (UInt64 &) const
- void convert (bool &value) const
- void convert (float &) const
- void convert (double &) const
- void convert (char &) const
- void convert (std::string &s) const
- void convert (DateTime &) const
- void convert (LocalDateTime &) const
- void convert (Timestamp &) const
- DynamicAnyHolder * clone () const
- const Util::JSONArray::Ptr & value () const
- bool isArray () const
- bool isInteger () const
- bool isSigned () const
- bool isNumeric () const
- bool isString () const

7.18.1 Constructor & Destructor Documentation

7.18.1.1 Poco::DynamicAnyHolderImpl< Util::JSONArray::Ptr >::DynamicAnyHolderImpl (const Util::JSONArray::Ptr & *val*) [inline]

7.18.1.2 Poco::DynamicAnyHolderImpl< Util::JSONArray::Ptr >::~DynamicAnyHolderImpl ( ) [inline]

7.18.2 Member Function Documentation

7.18.2.1 DynamicAnyHolder* Poco::DynamicAnyHolderImpl< Util::JSONArray::Ptr >::clone ( ) const [inline]

7.18.2.2 void Poco::DynamicAnyHolderImpl< Util::JSONArray::Ptr >::convert (Int8 &) const [inline]

7.18.2.3 void Poco::DynamicAnyHolderImpl< Util::JSONArray::Ptr >::convert (Int16 &) const [inline]

7.18.2.4 void Poco::DynamicAnyHolderImpl< Util::JSONArray::Ptr >::convert (Int32 &) const [inline]

7.18.2.5 void Poco::DynamicAnyHolderImpl< Util::JSONArray::Ptr >::convert (Int64 &) const [inline]

7.18.2.6 void Poco::DynamicAnyHolderImpl< Util::JSONArray::Ptr >::convert (UInt8 &) const [inline]

7.18.2.7 void Poco::DynamicAnyHolderImpl< Util::JSONArray::Ptr >::convert (UInt16 &) const [inline]

7.18.2.8 void Poco::DynamicAnyHolderImpl< Util::JSONArray::Ptr >::convert (UInt32 &) const [inline]

7.18.2.9 void Poco::DynamicAnyHolderImpl< Util::JSONArray::Ptr >::convert (UInt64 &) const [inline]

7.18.2.10 void Poco::DynamicAnyHolderImpl< Util::JSONArray::Ptr >::convert (bool & *value*) const [inline]

7.18.2.11 void Poco::DynamicAnyHolderImpl< Util::JSONArray::Ptr >::convert (float &) const [inline]

7.18.2.12 void Poco::DynamicAnyHolderImpl< Util::JSONArray::Ptr >::convert (double &) const [inline]

7.18.2.13 void Poco::DynamicAnyHolderImpl< Util::JSONArray::Ptr >::convert (char &) const [inline]

7.18.2.14 void Poco::DynamicAnyHolderImpl< Util::JSONArray::Ptr >::convert (std::string & *s*) const [inline]

7.18.2.15 void Poco::DynamicAnyHolderImpl< Util::JSONArray::Ptr >::convert (DateTime &) const [inline]

7.18.2.16 void Poco::DynamicAnyHolderImpl< Util::JSONArray::Ptr >::convert (LocalDateTime &) const [inline]

7.18.2.17 void Poco::DynamicAnyHolderImpl< Util::JSONArray::Ptr >::convert (Timestamp &) const [inline]

7.18.2.18 bool Poco::DynamicAnyHolderImpl< Util::JSONArray::Ptr >::isArray ( ) const [inline]

7.18.2.19 bool Poco::DynamicAnyHolderImpl< Util::JSONArray::Ptr >::isInteger ( ) const [inline]

7.18.2.20 bool Poco::DynamicAnyHolderImpl< Util::JSONArray::Ptr >::isNumeric ( ) const [inline]

7.18.2.21 bool Poco::DynamicAnyHolderImpl< Util::JSONArray::Ptr >::isSigned ( ) const [inline]

7.18.2.22 bool Poco::DynamicAnyHolderImpl< Util::JSONArray::Ptr >::isString ( ) const [inline]

7.18.2.23 const std::type_info& Poco::DynamicAnyHolderImpl< Util::JSONArray::Ptr >::type ( ) const [inline]

7.18.2.24 const Util::JSONArray::Ptr& Poco::DynamicAnyHolderImpl< Util::JSONArray::Ptr >::value ( ) const [inline]

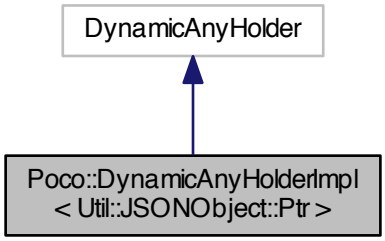
The documentation for this class was generated from the following file:

- /Users/rakesh/projects/opensource/mongoviewer/bson/src/api/Poco/Util/JSONArray.h

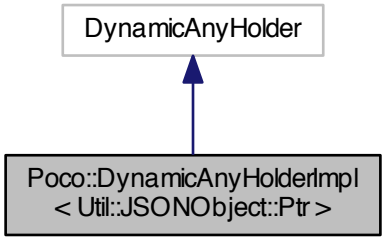
7.19 Poco::DynamicAnyHolderImpl< Util::JSONObject::Ptr > Class Template Reference

`#include <JSONObject.h>`

Inheritance diagram for Poco::DynamicAnyHolderImpl< Util::JSONObject::Ptr >:



Collaboration diagram for Poco::DynamicAnyHolderImpl< Util::JSONObject::Ptr >:



Public Member Functions

- DynamicAnyHolderImpl (const Util::JSONObject::Ptr &val)
- ~DynamicAnyHolderImpl ()
- const std::type_info & type () const
- void convert (Int8 &) const
- void convert (Int16 &) const
- void convert (Int32 &) const
- void convert (Int64 &) const
- void convert (UInt8 &) const
- void convert (UInt16 &) const
- void convert (UInt32 &) const
- void convert (UInt64 &) const
- void convert (bool &value) const
- void convert (float &) const
- void convert (double &) const
- void convert (char &) const
- void convert (std::string &s) const
- void convert (DateTime &) const
- void convert (LocalDateTime &) const
- void convert (Timestamp &) const
- DynamicAnyHolder * clone () const
- const Util::JSONObject::Ptr & value () const
- bool isArray () const
- bool isInteger () const
- bool isSigned () const
- bool isNumeric () const
- bool isString () const

7.19.1 Constructor & Destructor Documentation

7.19.1.1 Poco::DynamicAnyHolderImpl< Util::JSONObject::Ptr >::DynamicAnyHolderImpl (const Util::JSONObject::Ptr & *val*) [inline]

7.19.1.2 Poco::DynamicAnyHolderImpl< Util::JSONObject::Ptr >::~DynamicAnyHolderImpl ( ) [inline]

7.19.2 Member Function Documentation

7.19.2.1 DynamicAnyHolder* Poco::DynamicAnyHolderImpl< Util::JSONObject::Ptr >::clone ( ) const [inline]

7.19.2.2 void Poco::DynamicAnyHolderImpl< Util::JSONObject::Ptr >::convert (Int8 &) const [inline]

7.19.2.3 void Poco::DynamicAnyHolderImpl< Util::JSONObject::Ptr >::convert (Int16 &) const [inline]

7.19.2.4 void Poco::DynamicAnyHolderImpl< Util::JSONObject::Ptr >::convert (Int32 &) const [inline]

7.19.2.5 void Poco::DynamicAnyHolderImpl< Util::JSONObject::Ptr >::convert (Int64 &) const [inline]

7.19.2.6 void Poco::DynamicAnyHolderImpl< Util::JSONObject::Ptr >::convert (UInt8 &) const [inline]

7.19.2.7 void Poco::DynamicAnyHolderImpl< Util::JSONObject::Ptr >::convert (UInt16 &) const [inline]

7.19.2.8 void Poco::DynamicAnyHolderImpl< Util::JSONObject::Ptr >::convert (UInt32 &) const [inline]

7.19.2.9 void Poco::DynamicAnyHolderImpl< Util::JSONObject::Ptr >::convert (UInt64 &) const [inline]

7.19.2.10 void Poco::DynamicAnyHolderImpl< Util::JSONObject::Ptr >::convert (bool & *value*) const [inline]

7.19.2.11 void Poco::DynamicAnyHolderImpl< Util::JSONObject::Ptr >::convert (float &) const [inline]

7.19.2.12 void Poco::DynamicAnyHolderImpl< Util::JSONObject::Ptr >::convert (double &) const [inline]

7.19.2.13 void Poco::DynamicAnyHolderImpl< Util::JSONObject::Ptr >::convert (char &) const [inline]

7.19.2.14 void Poco::DynamicAnyHolderImpl< Util::JSONObject::Ptr >::convert (std::string & *s*) const [inline]

7.19.2.15 void Poco::DynamicAnyHolderImpl< Util::JSONObject::Ptr >::convert (DateTime &) const [inline]

7.19.2.16 void Poco::DynamicAnyHolderImpl< Util::JSONObject::Ptr >::convert (LocalDateTime &) const [inline]

7.19.2.17 void Poco::DynamicAnyHolderImpl< Util::JSONObject::Ptr >::convert (Timestamp &) const [inline]

7.19.2.18 bool Poco::DynamicAnyHolderImpl< Util::JSONObject::Ptr >::isArray ( ) const [inline]

7.19.2.19 bool Poco::DynamicAnyHolderImpl< Util::JSONObject::Ptr >::isInteger ( ) const [inline]

7.19.2.20 bool Poco::DynamicAnyHolderImpl< Util::JSONObject::Ptr >::isNumeric ( ) const [inline]

7.19.2.21 bool Poco::DynamicAnyHolderImpl< Util::JSONObject::Ptr >::isSigned ( ) const [inline]

7.19.2.22 bool Poco::DynamicAnyHolderImpl< Util::JSONObject::Ptr >::isString ( ) const [inline]

7.19.2.23 const std::type_info& Poco::DynamicAnyHolderImpl< Util::JSONObject::Ptr >::type ( ) const [inline]

7.19.2.24 const Util::JSONObject::Ptr& Poco::DynamicAnyHolderImpl< Util::JSONObject::Ptr >::value ( ) const [inline]

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

- /Users/rakesh/projects/opensource/mongoviewer/bson/src/api/Poco/Util/JSONObject.h

7.20 uma::bson::Element Class Reference

A class that represents a BSON element. Stores the name-value mapping for a BSON element in a PIMPL. This enables efficient copy-by-value semantics at the cost of shared data across the various copies of the element.

`#include <Element.h>`

Public Member Functions

- Element ()

  Default CTOR.
- template<typename DataType >
  Element (const std::string &fieldName, const DataType &v)

  Create a new element with the specified name and value.
- const std::string & getName () const

  Return the name of the element.
- Element & setName (const std::string &name)

  Rename this element.
- template<typename DataType >
  DataType & getValue ()

  Return a reference to the value held in this element for editing.
- template<typename DataType >
  const DataType & getValue () const

  Return a constant reference to the value held in this element.
- template<typename DataType >
  Element & setValue (const DataType &dt)

  Set the value held in this element to the specified value.
- Element & set (Value *val)

  Set the value held in this element to the specified value.
- template<typename DataType >
  Element & operator= (DataType value)

  Copy assignment operator used to assign a new value to this element.
- template<typename SimpleType >
  SimpleType getSimple () const

  A convenience method to retrieve the value of a simple type. This cannot be implemented as a template specialisation of the getValue() method since those methods return references to the value.
- template<typename DataType >
  Element & setSimple (const DataType)

  Set the value for this element. This is supported only for the specialisations provided (primarily for setting simple primitive type values).
- int32_t getSize () const

  Returns the total size in bytes the BSON representation of this element.
- Value::Type getType () const

  Returns the type for data stored in this element.
- template<>
  UMA_BSON_API Element (const std::string &name, const double &value)

  Specialised CTOR for an element holding a uma::bson::Double.
- template<>
  UMA_BSON_API Element (const std::string &name, const std::string &value)

  Specialised CTOR for an element holding a uma::bson::String.
- template<>
  UMA_BSON_API Element (const std::string &name, const bool &value)

  Specialised CTOR for an element holding a uma::bson::Boolean.

- template<>
  UMA_BSON_API Element (const std::string &name, const int &value)

  Specialised CTOR for an element holding a uma::bson::Integer.
- template<>
  UMA_BSON_API Element (const std::string &name, const int64_t &value)

  Specialised CTOR for an element holding a uma::bson::Long.
- template<>
  UMA_BSON_API Element & setValue (const double &v)

  Specialised method for setting the value to uma::bson::Double.
- template<>
  UMA_BSON_API Element & setValue (const bool &v)

  Specialised method for setting the value to uma::bson::Boolean.
- template<>
  UMA_BSON_API Element & setValue (const int32_t &v)

  Specialised method for setting the value to uma::bson::Integer.
- template<>
  UMA_BSON_API Element & setValue (const int64_t &v)

  Specialised method for setting the value to uma::bson::Long.
- template<>
  UMA_BSON_API Element & operator= (const double value)

  Specialised method for setting the value to uma::bson::Double.
- template<>
  UMA_BSON_API Element & operator= (const std::string &value)

  Specialised method for setting the value to uma::bson::String.
- template<>
  UMA_BSON_API Element & operator= (const char *value)

  Specialised method for setting the value to uma::bson::String.
- template<>
  UMA_BSON_API Element & operator= (const bool value)

  Specialised method for setting the value to uma::bson::Boolean.
- template<>
  UMA_BSON_API Element & operator= (const int32_t value)

  Specialised method for setting the value to uma::bson::Integer.
- template<>
  UMA_BSON_API Element & operator= (const int64_t value)

  Specialised method for setting the value to uma::bson::Long.

Static Public Member Functions

- static Element createElement (const std::string &name, Value *value)

  Create a new element with the specified name and value.

7.20.1 Detailed Description

A class that represents a BSON element. Stores the name-value mapping for a BSON element in a PIMPL. This enables efficient copy-by-value semantics at the cost of shared data across the various copies of the element.

Date

Created 2012/09/05 19:49

Copyright

Author

Rakesh

Version

Id:

Element.h 203 2013-02-15 12:23:15Z spt

7.20.2 Constructor & Destructor Documentation

7.20.2.1 uma::bson::Element::Element ( ) [inline]

Default CTOR.

7.20.2.2 template<typename DataType > uma::bson::Element::Element (const std::string & *fieldName,* const DataType & *v*) [inline]

Create a new element with the specified name and value.

Template Parameters

DataType	The type of value to store in the element.

Parameters

fieldName	The field name for this element.
v	The value for the element

7.20.2.3 template<> UMA_BSON_API uma::bson::Element::Element (const std::string & *name,* const double & *value*)

Specialised CTOR for an element holding a uma::bson::Double.

7.20.2.4 template<> UMA_BSON_API uma::bson::Element::Element (const std::string & *name,* const std::string & *value*)

Specialised CTOR for an element holding a uma::bson::String.

7.20.2.5 template<> UMA_BSON_API uma::bson::Element::Element (const std::string & *name,* const bool & *value*)

Specialised CTOR for an element holding a uma::bson::Boolean.

7.20.2.6 template<> UMA_BSON_API uma::bson::Element::Element (const std::string & *name,* const int & *value*)

Specialised CTOR for an element holding a uma::bson::Integer.

7.20.2.7 template<> UMA_BSON_API uma::bson::Element::Element (const std::string & *name*, const int64_t & *value*)

Specialised CTOR for an element holding a uma::bson::Long.

7.20.3 Member Function Documentation

7.20.3.1 static Element uma::bson::Element::createElement (const std::string & *name*, Value * *value*) [static]

Create a new element with the specified name and value.

Warning

Note that this element takes ownership of the specified value instance. Callers must not `delete` the passed in value.

Since

Version 2.2

Parameters

name	The field name of the this element
value	The value of this element.

7.20.3.2 const std::string& uma::bson::Element::getName ( ) const [inline]

Return the name of the element.

Returns

The field name.

7.20.3.3 UMA_BSON_API int64_t uma::bson::Element::getSimple< int64_t > ( ) const [inline]

A convenience method to retrieve the value of a simple type. This cannot be implemented as a template specialisation of the getValue() method since those methods return references to the value.

Specialised method for getting the value of uma::bson::Long.

Specialised method for getting the value of uma::bson::Integer.

Specialised method for getting the value of uma::bson::Boolean.

Specialised method for getting the value of uma::bson::Double.

Template specialisations are provided for the common simple types (`double`, `string`, `bool`, `integer`, `long`) which will work as expected. This method makes it easier to retrieve simple values than the full syntax. For example:

```
const double dval = elem->getValue<Double>().getValue();
```

becomes

```
const double dval = elem->getSimple<double>();
```

Template Parameters

SimpleType	The primitive represented by the element value.

Returns

The simple value.

7.20.3.4 int32_t uma::bson::Element::getSize ( ) const [inline]

Returns the total size in bytes the BSON representation of this element.

Returns

The size in bytes

7.20.3.5 Value::Type uma::bson::Element::getType ( ) const [inline]

Returns the type for data stored in this element.

Returns

The current data type

7.20.3.6 template<typename DataType > DataType& uma::bson::Element::getValue ( ) [inline]

Return a reference to the value held in this element for editing.

If the template type specified does not match the current data type, or if the value is `null` an exception is thrown.

Template Parameters

DataType	The type of value stored in the element.

Returns

The current value held in this element.

Exceptions

Poco::NullValueException	If no value is currently held in this element.
Poco::BadCastException	If the template type specified does not match the type held in this element.

7.20.3.7 template<typename DataType > const DataType& uma::bson::Element::getValue ( ) const [inline]

Return a constant reference to the value held in this element.

If the template type specified does not match the current data type, or if the value is `null` an exception is thrown.

Template Parameters

DataType	The type of value stored in the element.

Returns

The current value held in this element.

Exceptions

Poco::NullValueException	If no value is currently held in this element.
Poco::BadCastException	If the template type specified does not match the type held in this element.

7.20.3.8 template<typename DataType > Element& uma::bson::Element::operator= (DataType *value*) [inline]

Copy assignment operator used to assign a new value to this element.

Warning

This element takes ownership of the passed in value instance. Callers must not `delete` the passed in value instance.

Since

Version 2.2

Template Parameters

DataType	The type of value stored in the element.

Parameters

value	The new value to associate with this element.

Returns

This instance for method chaining.

7.20.3.9 template<> UMA_BSON_API Element& uma::bson::Element::operator= (const double *value*)

Specialised method for setting the value to uma::bson::Double.

7.20.3.10 template<> UMA_BSON_API Element& uma::bson::Element::operator= (const std::string & *value*)

Specialised method for setting the value to uma::bson::String.

7.20.3.11 template<> UMA_BSON_API Element& uma::bson::Element::operator= (const char * *value*)

Specialised method for setting the value to uma::bson::String.

7.20.3.12 template<> UMA_BSON_API Element& uma::bson::Element::operator= (const bool *value*)

Specialised method for setting the value to uma::bson::Boolean.

7.20.3.13 template<> UMA_BSON_API Element& uma::bson::Element::operator= (const int32_t *value*)

Specialised method for setting the value to uma::bson::Integer.

7.20.3.14 template<> UMA_BSON_API Element& uma::bson::Element::operator= (const int64_t *value*)

Specialised method for setting the value to uma::bson::Long.

7.20.3.15 Element& uma::bson::Element::set (Value * *val*) [inline]

Set the value held in this element to the specified value.

Warning

Note that the element takes ownership of the specified value instance. Callers must not `delete` the passed in value.

Parameters

val	The value to set for this element.

Returns

This instance for method chaining.

7.20.3.16 Element& uma::bson::Element::setName (const std::string & *name*) [inline]

Rename this element.

Warning

Note that this is an internal use only method. If you invoke this method from client code, the element may become invisible to the containing document. Client's should always use the methods in the document/array classes to manage elements, which in turn will invoke this method to update the instance as appropriate.

Parameters

name	The new name for this element.

7.20.3.17 template<typename DataType > Element& uma::bson::Element::setSimple (const DataType) [inline]

Set the value for this element. This is supported only for the specialisations provided (primarily for setting simple primitive type values).

Template Parameters

DataType	The primitive value type.

Parameters

v	The value to set

Returns

This instance for method chaining.

7.20.3.18 template<typename DataType > Element& uma::bson::Element::setValue (const DataType & *dt*) [inline]

Set the value held in this element to the specified value.

Template Parameters

DataType	The type of value stored in the element.

Parameters

dt	The value to set for this element.

Returns

This instance for method chaining.

7.20.3.19 template<> UMA_BSON_API Element& uma::bson::Element::setValue (const double & *v*)

Specialised method for setting the value to uma::bson::Double.

7.20.3.20 template<> UMA_BSON_API Element& uma::bson::Element::setValue (const bool & *v*)

Specialised method for setting the value to uma::bson::Boolean.

7.20.3.21 template<> UMA_BSON_API Element& uma::bson::Element::setValue (const int32_t & *v*)

Specialised method for setting the value to uma::bson::Integer.

7.20.3.22 template<> UMA_BSON_API Element& uma::bson::Element::setValue (const int64_t & *v*)

Specialised method for setting the value to uma::bson::Long.

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

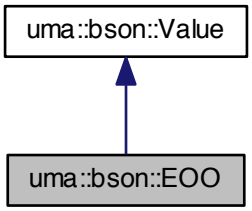
- /Users/rakesh/projects/opensource/mongoviewer/bson/src/api/uma/bson/Element.h

7.21 uma::bson::EOO Class Reference

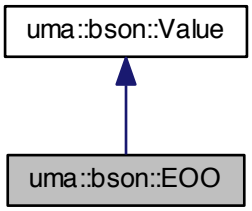
A POD that represents a EOO value.

`#include <EOO.h>`

Inheritance diagram for uma::bson::EOO:



Collaboration diagram for uma::bson::EOO:



Public Member Functions

- EOO ()

  Default CTOR.
- int32_t getSize () const

  Returns the size of the bson representation of this value as per the BSON specifications.
- Value::Type getType () const

  Returns the type for this instance as listed in the BSON specifications.

Additional Inherited Members

7.21.1 Detailed Description

A POD that represents a EOO value.

Date

Created 2012/09/05 20:30

Copyright

Copyright ©2012, Sans Pareil Technologies, Inc.

Author

Rakesh

Version

Id:

EOO.h 181 2012-12-21 01:08:58Z spt

7.21.2 Constructor & Destructor Documentation

7.21.2.1 uma::bson::EOO::EOO ( ) [inline]

Default CTOR.

7.21.3 Member Function Documentation

7.21.3.1 int32_t uma::bson::EOO::getSize ( ) const [inline], [virtual]

Returns the size of the bson representation of this value as per the BSON specifications.

Returns

Returns 0

Implements uma::bson::Value.

7.21.3.2 Value::Type uma::bson::EOO::getType ( ) const [inline], [virtual]

Returns the type for this instance as listed in the BSON specifications.

Returns

Returns uma::bson::Value::Type::Eoo

Implements uma::bson::Value.

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

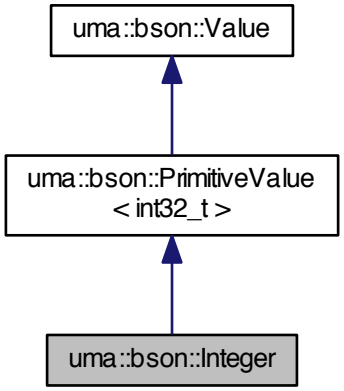
- /Users/rakesh/projects/opensource/mongoviewer/bson/src/api/uma/bson/EOO.h

7.22 uma::bson::Integer Class Reference

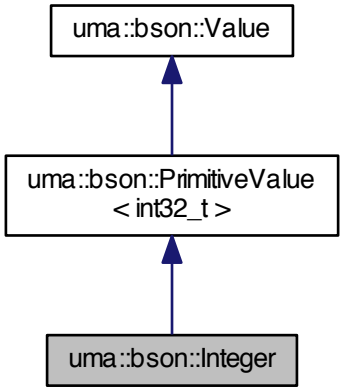
A POD that represents an integer type BSON element value.

`#include <Integer.h>`

Inheritance diagram for uma::bson::Integer:



Collaboration diagram for uma::bson::Integer:



Public Member Functions

- Integer ()

  Default CTOR.
- Integer (const int32_t v)

  Create a new instance that encapsulates the specified value.
- int32_t getSize () const

  Return the size in bytes that the encapsulated value represents as per the BSON specifications.
- Value::Type getType () const

  Returns the type for this instance as listed in the BSON specifications.

Additional Inherited Members

7.22.1 Detailed Description

A POD that represents an integer type BSON element value.

Date

Created 2012/09/13 18:02

Copyright

Author

Rakesh

Version

Id:

Integer.h 172 2012-12-14 17:21:21Z spt

7.22.2 Constructor & Destructor Documentation

7.22.2.1 uma::bson::Integer::Integer ( ) [inline]

Default CTOR.

7.22.2.2 uma::bson::Integer::Integer (const int32_t *v*) [inline], [explicit]

Create a new instance that encapsulates the specified value.

Parameters

v	The integer to encapsulate

7.22.3 Member Function Documentation

7.22.3.1 int32_t uma::bson::Integer::getSize ( ) const [inline], [virtual]

Return the size in bytes that the encapsulated value represents as per the BSON specifications.

Returns

Returns `4` as per specifications.

Implements uma::bson::Value.

7.22.3.2 Value::Type uma::bson::Integer::getType ( ) const [inline], [virtual]

Returns the type for this instance as listed in the BSON specifications.

Returns

Returns uma::bson::Value::Type::Integer

Implements uma::bson::Value.

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

- /Users/rakesh/projects/opensource/mongoviewer/bson/src/api/uma/bson/Integer.h

7.23 Poco::Util::JSONArray Class Reference

`#include <JSONArray.h>`

Public Types

- typedef SharedPtr< JSONArray > Ptr

Public Member Functions

- JSONArray ()
  Default constructor.
- JSONArray (const JSONArray &copy)
  Copy Constructor.
- virtual ~JSONArray ()
  Destructor.
- DynamicAny get (unsigned int index) const
  Retrieves an element. Will return an empty value when the element doesn't exist.
- JSONArray::Ptr getArray (unsigned int index) const
  Retrieves an array. When the element is not an array or doesn't exist, an empty SharedPtr is returned.
- template<typename T >
  T getElement (unsigned int index) const
  Retrieves an element and tries to convert it to the template type. The convert<T> method of Dynamic is called which can also throw exceptions for invalid values. Note: This will not work for an array or an object.
- SharedPtr< JSONObject > getObject (unsigned int index) const
  Retrieves an object. When the element is not an object or doesn't exist, an empty SharedPtr is returned.
- unsigned int size () const
  Returns the size of the array.
- bool isArray (unsigned int index) const
  Returns true when the element is an array.
- bool isNull (unsigned int index) const
  Returns true when the element is null or when the element doesn't exist.
- bool isObject (unsigned int index) const
  Returns true when the element is an object.
- template<typename T >
  T optElement (unsigned int index, const T &def) const
  Returns the element at the given index. When the element is null, doesn't exist or can't be converted to the given type, the default value will be returned.
- void add (const DynamicAny &value)
  Add the given value to the array.
- void stringify (std::ostream &out, unsigned int indent) const
  Prints the array to out. When indent is 0, the array will be printed on one line without indentation.
- void remove (unsigned int index)
  Removes the element on the given index.

7.23.1 Member Typedef Documentation

7.23.1.1 typedef SharedPtr<JSONArray> Poco::Util::JSONArray::Ptr

7.23.2 Constructor & Destructor Documentation

7.23.2.1 Poco::Util::JSONArray::JSONArray ( )

Default constructor.

7.23.2.2 Poco::Util::JSONArray::JSONArray (const JSONArray & *copy*)

Copy Constructor.

7.23.2.3 virtual Poco::Util::JSONArray::~JSONArray ( ) [virtual]

Destructor.

7.23.3 Member Function Documentation

7.23.3.1 void Poco::Util::JSONArray::add (const DynamicAny & *value*) [inline]

Add the given value to the array.

7.23.3.2 DynamicAny Poco::Util::JSONArray::get (unsigned int *index*) const

Retrieves an element. Will return an empty value when the element doesn't exist.

7.23.3.3 JSONArray::Ptr Poco::Util::JSONArray::getArray (unsigned int *index*) const

Retrieves an array. When the element is not an array or doesn't exist, an empty SharedPtr is returned.

7.23.3.4 template<typename T > T Poco::Util::JSONArray::getElement (unsigned int *index*) const [inline]

Retrieves an element and tries to convert it to the template type. The convert<T> method of Dynamic is called which can also throw exceptions for invalid values. Note: This will not work for an array or an object.

7.23.3.5 SharedPtr<JSONObject> Poco::Util::JSONArray::getObject (unsigned int *index*) const

Retrieves an object. When the element is not an object or doesn't exist, an empty SharedPtr is returned.

7.23.3.6 bool Poco::Util::JSONArray::isArray (unsigned int *index*) const [inline]

Returns true when the element is an array.

7.23.3.7 bool Poco::Util::JSONArray::isNull (unsigned int *index*) const [inline]

Returns true when the element is null or when the element doesn't exist.

7.23.3.8 bool Poco::Util::JSONArray::isObject (unsigned int *index*) const

Returns true when the element is an object.

7.23.3.9 template<typename T > T Poco::Util::JSONArray::optElement (unsigned int *index*, const T & *def*) const [inline]

Returns the element at the given index. When the element is null, doesn't exist or can't be converted to the given type, the default value will be returned.

7.23.3.10 void Poco::Util::JSONArray::remove (unsigned int *index*) [inline]

Removes the element on the given index.

7.23.3.11 unsigned int Poco::Util::JSONArray::size ( ) const [inline]

Returns the size of the array.

7.23.3.12 void Poco::Util::JSONArray::stringify (std::ostream & *out*, unsigned int *indent*) const

Prints the array to out. When indent is 0, the array will be printed on one line without indentation.

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

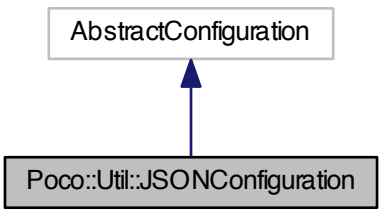
- /Users/rakesh/projects/opensource/mongoviewer/bson/src/api/Poco/Util/JSONArray.h

7.24 Poco::Util::JSONConfiguration Class Reference

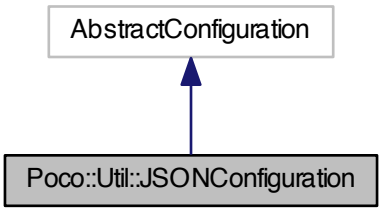
This configuration class extracts configuration properties from a JSON object. An XPath-like syntax for property names is supported to allow full access to the JSON object.

`#include <JSONConfiguration.h>`

Inheritance diagram for Poco::Util::JSONConfiguration:



Collaboration diagram for Poco::Util::JSONConfiguration:



Public Member Functions

- JSONConfiguration ()
- JSONConfiguration (const std::string &path)

  Creates an empty configuration.
- JSONConfiguration (std::istream &istr)

  Creates a configuration and loads the JSON structure from the given stream.
- JSONConfiguration (const JSONObject::Ptr &object)

  Creates a configuration from the given JSON object.
- virtual ~JSONConfiguration ()

  Destructor.
- void load (const std::string &path)

  Loads the configuration from the given file.
- void load (std::istream &istr)

  Loads the configuration from the given stream.
- void loadEmpty (const std::string &root)

  Loads an empty object containing only a root object with the given name.
- void save (std::ostream &ostr, unsigned int indent=2) const

  Saves the configuration to the given stream.
- void setInt (const std::string &key, int value)
- void setBool (const std::string &key, bool value)
- void setDouble (const std::string &key, double value)

Protected Member Functions

- bool getRaw (const std::string &key, std::string &value) const
- void setRaw (const std::string &key, const std::string &value)
- void enumerate (const std::string &key, Keys &range) const

7.24.1 Detailed Description

This configuration class extracts configuration properties from a JSON object. An XPath-like syntax for property names is supported to allow full access to the JSON object.

Given the following JSON object as an example: { "config" : { "prop1" : "value1", "prop2" : 10, "prop3" : ["element1", "element2"], "prop4" : { "prop5" : false, "prop6" : null } } } The following property names would be valid and would yield the shown values:

config.prop1 –> "value1" config.prop3[1] –> "element2" config.prop4.prop5 –> false

7.24.2 Constructor & Destructor Documentation

7.24.2.1 Poco::Util::JSONConfiguration::JSONConfiguration ( )

7.24.2.2 Poco::Util::JSONConfiguration::JSONConfiguration (const std::string & *path*)

Creates an empty configuration.

Creates a configuration and loads the JSON structure from the given file

7.24.2.3 Poco::Util::JSONConfiguration::JSONConfiguration (std::istream & *istr*)

Creates a configuration and loads the JSON structure from the given stream.

7.24.2.4 Poco::Util::JSONConfiguration::JSONConfiguration (const **JSONObject::Ptr** & *object*)

Creates a configuration from the given JSON object.

7.24.2.5 virtual Poco::Util::JSONConfiguration::~JSONConfiguration ( ) [virtual]

Destructor.

7.24.3 Member Function Documentation

7.24.3.1 void Poco::Util::JSONConfiguration::enumerate (const std::string & *key,* Keys & *range*) const [protected]

7.24.3.2 bool Poco::Util::JSONConfiguration::getRaw (const std::string & *key,* std::string & *value*) const [protected]

7.24.3.3 void Poco::Util::JSONConfiguration::load (const std::string & *path*)

Loads the configuration from the given file.

7.24.3.4 void Poco::Util::JSONConfiguration::load (std::istream & *istr*)

Loads the configuration from the given stream.

7.24.3.5 void Poco::Util::JSONConfiguration::loadEmpty (const std::string & *root*)

Loads an empty object containing only a root object with the given name.

7.24.3.6 void Poco::Util::JSONConfiguration::save (std::ostream & *ostr,* unsigned int *indent* = 2) const

Saves the configuration to the given stream.

7.24.3.7 void Poco::Util::JSONConfiguration::setBool (const std::string & *key,* bool *value*)

7.24.3.8 void Poco::Util::JSONConfiguration::setDouble (const std::string & *key,* double *value*)

7.24.3.9 void Poco::Util::JSONConfiguration::setInt (const std::string & *key,* int *value*)

7.24.3.10 void Poco::Util::JSONConfiguration::setRaw (const std::string & *key*, const std::string & *value*) [protected]

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

- /Users/rakesh/projects/opensource/mongoviewer/bson/src/api/Poco/Util/JSONConfiguration.h

7.25 Poco::Util::JSONObject Class Reference

Represents a JSON object.

`#include <JSONObject.h>`

Public Types

- typedef SharedPtr< JSONObject > Ptr

Public Member Functions

- JSONObject ()

  Default constructor.
- JSONObject (const JSONObject &copy)
- virtual ~JSONObject ()

  Destructor.
- DynamicAny get (const std::string &key) const

  Retrieves a property. An empty value is returned when the property doesn't exist.
- JSONArray::Ptr getArray (const std::string &key) const

  Returns a SharedPtr to an array when the property is an array. An empty SharedPtr is returned when the element doesn't exist or is not an array.
- JSONObject::Ptr getObject (const std::string &key) const

  Returns a SharedPtr to an object when the property is an object. An empty SharedPtr is returned when the property doesn't exist or is not an object.
- template<typename T >
  T getValue (const std::string &key) const

  Retrieves the property with the given name and will try to convert the value to the given template type. The convert<-T> method of Dynamic is called which can also throw exceptions for invalid values. Note: This will not work for an array or an object.
- void getNames (std::vector< std::string > &names) const

  Returns all property names.
- bool has (const std::string &key) const

  Returns true when the given property exists.
- bool isArray (const std::string &key) const

  Returns true when the given property contains an array.
- bool isNull (const std::string &key) const

  Returns true when the given property contains a null value.
- bool isObject (const std::string &key) const

  Returns true when the given property contains an object.
- template<typename T >
  T optValue (const std::string &key, const T &def) const

  Returns the value of a property when the property exists and can be converted to the given type. Otherwise def will be returned.
- unsigned int size () const

  Returns the number of properties.

- void set (const std::string &key, const DynamicAny &value)

  Sets a new value.
- void stringify (std::ostream &out, unsigned int indent=0) const

  Prints the object to out. When indent is 0, the object will be printed on one line without indentation.
- void remove (const std::string &key)

  Removes the property with the given key.

7.25.1 Detailed Description

Represents a JSON object.

7.25.2 Member Typedef Documentation

7.25.2.1 typedef SharedPtr<JSONObject> Poco::Util::JSONObject::Ptr

7.25.3 Constructor & Destructor Documentation

7.25.3.1 Poco::Util::JSONObject::JSONObject ( )

Default constructor.

7.25.3.2 Poco::Util::JSONObject::JSONObject (const JSONObject & *copy*)

7.25.3.3 virtual Poco::Util::JSONObject::~JSONObject ( ) [virtual]

Destructor.

7.25.4 Member Function Documentation

7.25.4.1 DynamicAny Poco::Util::JSONObject::get (const std::string & *key*) const

Retrieves a property. An empty value is returned when the property doesn't exist.

7.25.4.2 JSONArray::Ptr Poco::Util::JSONObject::getArray (const std::string & *key*) const

Returns a SharedPtr to an array when the property is an array. An empty SharedPtr is returned when the element doesn't exist or is not an array.

7.25.4.3 void Poco::Util::JSONObject::getNames (std::vector< std::string > & *names*) const

Returns all property names.

7.25.4.4 JSONObject::Ptr Poco::Util::JSONObject::getObject (const std::string & *key*) const

Returns a SharedPtr to an object when the property is an object. An empty SharedPtr is returned when the property doesn't exist or is not an object.

7.25.4.5 template<typename T > T Poco::Util::JSONObject::getValue (const std::string & *key*) const [inline]

Retrieves the property with the given name and will try to convert the value to the given template type. The convert<-T> method of Dynamic is called which can also throw exceptions for invalid values. Note: This will not work for an array or an object.

7.25.4.6 bool Poco::Util::JSONObject::has (const std::string & *key*) const [inline]

Returns true when the given property exists.

7.25.4.7 bool Poco::Util::JSONObject::isArray (const std::string & *key*) const [inline]

Returns true when the given property contains an array.

7.25.4.8 bool Poco::Util::JSONObject::isNull (const std::string & *key*) const [inline]

Returns true when the given property contains a null value.

7.25.4.9 bool Poco::Util::JSONObject::isObject (const std::string & *key*) const [inline]

Returns true when the given property contains an object.

7.25.4.10 template<typename T > T Poco::Util::JSONObject::optValue (const std::string & *key,* const T & *def*) const [inline]

Returns the value of a property when the property exists and can be converted to the given type. Otherwise def will be returned.

7.25.4.11 void Poco::Util::JSONObject::remove (const std::string & *key*) [inline]

Removes the property with the given key.

7.25.4.12 void Poco::Util::JSONObject::set (const std::string & *key,* const DynamicAny & *value*) [inline]

Sets a new value.

7.25.4.13 unsigned int Poco::Util::JSONObject::size ( ) const [inline]

Returns the number of properties.

7.25.4.14 void Poco::Util::JSONObject::stringify (std::ostream & *out,* unsigned int *indent* = 0) const

Prints the object to out. When indent is 0, the object will be printed on one line without indentation.

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

- /Users/rakesh/projects/opensource/mongoviewer/bson/src/api/Poco/Util/JSONObject.h

7.26 Poco::Util::JSONParser Class Reference

A class for passing JSON strings or streams.

`#include <JSONParser.h>`

Public Member Functions

- JSONParser ()
  Constructor.
- virtual ~JSONParser ()
  Destructor.
- DynamicAny parse (const std::string &source)
  Parses a string.
- DynamicAny parse (std::istream &in)
  Parses a JSON from the input stream.

7.26.1 Detailed Description

A class for passing JSON strings or streams.

7.26.2 Constructor & Destructor Documentation

7.26.2.1 Poco::Util::JSONParser::JSONParser ( )

Constructor.

7.26.2.2 virtual Poco::Util::JSONParser::~JSONParser ( ) [virtual]

Destructor.

7.26.3 Member Function Documentation

7.26.3.1 DynamicAny Poco::Util::JSONParser::parse (const std::string & *source*) [inline]

Parses a string.

7.26.3.2 DynamicAny Poco::Util::JSONParser::parse (std::istream & *in*)

Parses a JSON from the input stream.

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

- /Users/rakesh/projects/opensource/mongoviewer/bson/src/api/Poco/Util/JSONParser.h

7.27 Poco::Util::JSONQuery Class Reference

Class that can be used to search for a value in a JSON object or array.

`#include <JSONQuery.h>`

Public Member Functions

- JSONQuery (const DynamicAny &source)

  Constructor. Pass the start object/array.
- virtual ~JSONQuery ()

  Destructor.
- JSONObject::Ptr findObject (const std::string &path) const

  Search for an object. When the object can't be found, an empty SharedPtr is returned.
- JSONArray::Ptr findArray (const std::string &path) const

  Search for an array. When the array can't be found, an empty SharedPtr is returned.
- DynamicAny find (const std::string &path) const

  Searches a value For example: "person.children[0].name" will return the the name of the first child. When the value can't be found an empty value is returned.
- template<typename T >
  T findValue (const std::string &path, const T &def) const

  Searches for a value will convert it to the given type. When the value can't be found or has an invalid type the default value will be returned.
- std::string findValue (const char *path, const char *def) const

  Searches for a value will convert it to the given type. When the value can't be found or has an invalid type the default value will be returned.

7.27.1 Detailed Description

Class that can be used to search for a value in a JSON object or array.

7.27.2 Constructor & Destructor Documentation

7.27.2.1 Poco::Util::JSONQuery::JSONQuery (const DynamicAny & *source*)

Constructor. Pass the start object/array.

7.27.2.2 virtual Poco::Util::JSONQuery::~JSONQuery ( ) [virtual]

Destructor.

7.27.3 Member Function Documentation

7.27.3.1 DynamicAny Poco::Util::JSONQuery::find (const std::string & *path*) const

Searches a value For example: "person.children[0].name" will return the the name of the first child. When the value can't be found an empty value is returned.

7.27.3.2 JSONArray::Ptr Poco::Util::JSONQuery::findArray (const std::string & *path*) const

Search for an array. When the array can't be found, an empty SharedPtr is returned.

7.27.3.3 JSONObject::Ptr Poco::Util::JSONQuery::findObject (const std::string & *path*) const

Search for an object. When the object can't be found, an empty SharedPtr is returned.

7.27.3.4 template<typename T > T Poco::Util::JSONQuery::findValue (const std::string & *path*, const T & *def*) const [inline]

Searches for a value will convert it to the given type. When the value can't be found or has an invalid type the default value will be returned.

7.27.3.5 std::string Poco::Util::JSONQuery::findValue (const char * *path*, const char * *def*) const [inline]

Searches for a value will convert it to the given type. When the value can't be found or has an invalid type the default value will be returned.

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

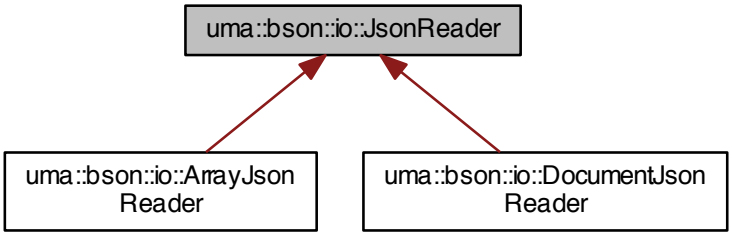
- /Users/rakesh/projects/opensource/mongoviewer/bson/src/api/Poco/Util/JSONQuery.h

7.28 uma::bson::io::JsonReader Class Reference

A base streaming parser that parses a stream of JSON bytes from an input stream and transforms into a object model.

`#include <JsonReader.h>`

Inheritance diagram for uma::bson::io::JsonReader:



Public Member Functions

- virtual ~JsonReader ()

  Virtual destructor for sub-classes.

Protected Member Functions

- Document & read (Poco::Util::JSONObject::Ptr object, Document &document)

  Read a JSON object structure into a document. Used primarily to read embedded documents.
- Array read (Poco::Util::JSONArray::Ptr arrayPtr)

  Read a JSON array structure into a array. Used primarily to read embedded arrays.
- BinaryData readBinary (Poco::Util::JSONObject::Ptr value)

  Read a nested binary data object from the current element.
- RegularExpression readRegex (Poco::Util::JSONObject::Ptr value)

  Read a nested regular expression object from the current element.

- DatabaseReference readDbRef (Poco::Util::JSONObject::Ptr value)

  Read a nested database reference object from the current element.
- CodeWithScope readCodeWScope (Poco::Util::JSONObject::Ptr value)

  Read a nested code with scope object from the current element.
- Timestamp readTimestamp (Poco::Util::JSONObject::Ptr value)

  Read a nested timestamp object from the current element.

7.28.1 Detailed Description

A base streaming parser that parses a stream of JSON bytes from an input stream and transforms into a object model.

Note that this parser will process only JSON data that was created in the format created by JsonWriter.

Copyright

Copyright ©2012, Sans Pareil Technologies, Inc.

Date

Created 2012/09/27 21:54

Author

Rakesh

Version

Id:

JsonReader.h 167 2012-12-13 22:11:47Z spt

7.28.2 Constructor & Destructor Documentation

7.28.2.1 virtual uma::bson::io::JsonReader::~JsonReader ( ) [inline], [virtual]

Virtual destructor for sub-classes.

7.28.3 Member Function Documentation

7.28.3.1 Document& uma::bson::io::JsonReader::read (Poco::Util::JSONObject::Ptr *object*, Document & *document*) [protected]

Read a JSON object structure into a document. Used primarily to read embedded documents.

Parameters

object	The embedded JSON object to read
document	The document that is to be popuated

Returns

Document The passed in document reference

7.28.3.2 Array uma::bson::io::JsonReader::read (Poco::Util::JSONArray::Ptr *arrayPtr*) [protected]

Read a JSON array structure into a array. Used primarily to read embedded arrays.

Parameters

object	The embedded JSON array to read

Returns

The parsed array instance.

7.28.3.3 BinaryData uma::bson::io::JsonReader::readBinary (Poco::Util::JSONObject::Ptr *value*) [protected]

Read a nested binary data object from the current element.

Parameters

value	The current element being processed

Returns

RegularExpression The encoded binary data

7.28.3.4 CodeWithScope uma::bson::io::JsonReader::readCodeWScope (Poco::Util::JSONObject::Ptr *value*) [protected]

Read a nested code with scope object from the current element.

Parameters

value	The current element being processed

Returns

RegularExpression The encoded code with scope

7.28.3.5 DatabaseReference uma::bson::io::JsonReader::readDbRef (Poco::Util::JSONObject::Ptr *value*) [protected]

Read a nested database reference object from the current element.

Parameters

value	The current element being processed

Returns

RegularExpression The encoded database reference

7.28.3.6 RegularExpression uma::bson::io::JsonReader::readRegex (Poco::Util::JSONObject::Ptr *value*) [protected]

Read a nested regular expression object from the current element.

Parameters

value	The current element being processed

Returns

RegularExpression The encoded regular expression

7.28.3.7 Timestamp uma::bson::io::JsonReader::readTimestamp (Poco::Util::JSONObject::Ptr *value*) [protected]

Read a nested timestamp object from the current element.

Parameters

value	The current element being processed

Returns

RegularExpression The encoded timestamp

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

- /Users/rakesh/projects/opensource/mongoviewer/bson/src/api/uma/bson/io/JsonReader.h

7.29 Poco::Util::JSONStringifier Class Reference

Helper class for creating a String from a JSON object or array.

`#include <JSONStringifier.h>`

Static Public Member Functions

- static void stringify (const DynamicAny &any, std::ostream &out, unsigned int indent=0)

  Writes a String representation of the value to the output stream. When indent is 0, the String will be created as small as possible.

7.29.1 Detailed Description

Helper class for creating a String from a JSON object or array.

7.29.2 Member Function Documentation

7.29.2.1 static void Poco::Util::JSONStringifier::stringify (const DynamicAny & *any*, std::ostream & *out*, unsigned int *indent* = 0) [static]

Writes a String representation of the value to the output stream. When indent is 0, the String will be created as small as possible.

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

- /Users/rakesh/projects/opensource/mongoviewer/bson/src/api/Poco/Util/JSONStringifier.h

7.30 Poco::Util::JSONTemplate Class Reference

JSONTemplate is a template engine which uses JSON as input for generating output. There are commands for looping over JSON arrays, include other templates, conditional output, ...

`#include <JSONTemplate.h>`

Public Types

- typedef SharedPtr< JSONTemplate > Ptr

Public Member Functions

- JSONTemplate ()

  Constructor.
- JSONTemplate (const Path &templatePath)

  Constructor. Creates a template from a file.
- virtual ~JSONTemplate ()

  Destructor.
- void parse ()

  Parse a template from a file.
- void parse (const std::string &source)

  Parse a template from a String.
- void parse (std::istream &in)

  Parse a template from a input stream.
- Timestamp parseTime () const

  Returns the time when the template was parsed.
- void render (const DynamicAny &data, std::ostream &out) const

  Renders the template and send the output to the stream.

7.30.1 Detailed Description

JSONTemplate is a template engine which uses JSON as input for generating output. There are commands for looping over JSON arrays, include other templates, conditional output, ...

All text is send to the outputstream. A command is placed between <? and ?>.

These are the available commands: <? echo query ?> The result of the query is send to the output stream This command can also be written as <?= query ?> <? if query ?> <? else ?> <? endif ?> When the result of query is true, all the text between if and else (or endif when there is no else) is send to the output stream. When the result of query is false, all the text between else and endif is send to the output stream. An empty object, an empty array or a null value is considered as a false value. For numbers a zero is false. An empty String is also false. <? ifexist query ?> <? else ?> <? endif ?> This can be used to check the existance of the value. Use this for example when a zero value is ok (which returns false for <? if ?>. <? for variable query ?> <? endfor ?> The result of the query must be an array. For each element in the array the text between for and endfor is send to the output stream. The active element is stored in the variable. <? include "filename" ?> Includes a template. When the filename is relative it will try to resolve the filename against the active template. When this file doesn't exist, it can still be found when the JSONTemplateCache is used.

A query is passed to JSONQuery to get the value.

7.30.2 Member Typedef Documentation

7.30.2.1 typedef SharedPtr<JSONTemplate> Poco::Util::JSONTemplate::Ptr

7.30.3 Constructor & Destructor Documentation

7.30.3.1 Poco::Util::JSONTemplate::JSONTemplate ( )

Constructor.

7.30.3.2 Poco::Util::JSONTemplate::JSONTemplate (const Path & *templatePath*)

Constructor. Creates a template from a file.

7.30.3.3 virtual Poco::Util::JSONTemplate::~JSONTemplate ( ) [virtual]

Destructor.

7.30.4 Member Function Documentation

7.30.4.1 void Poco::Util::JSONTemplate::parse ( )

Parse a template from a file.

7.30.4.2 void Poco::Util::JSONTemplate::parse (const std::string & *source*) [inline]

Parse a template from a String.

7.30.4.3 void Poco::Util::JSONTemplate::parse (std::istream & *in*)

Parse a template from a input stream.

7.30.4.4 Timestamp Poco::Util::JSONTemplate::parseTime ( ) const [inline]

Returns the time when the template was parsed.

7.30.4.5 void Poco::Util::JSONTemplate::render (const DynamicAny & *data,* std::ostream & *out*) const

Renders the template and send the output to the stream.

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

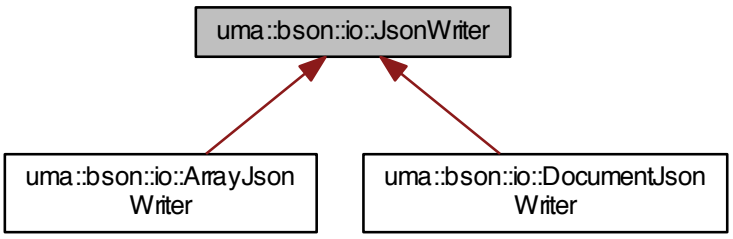
- /Users/rakesh/projects/opensource/mongoviewer/bson/src/api/Poco/Util/JSONTemplate.h

7.31 uma::bson::io::JsonWriter Class Reference

A streaming writer that transforms an object model into a JSON representation.

`#include <JsonWriter.h>`

Inheritance diagram for uma::bson::io::JsonWriter:



Public Member Functions

- virtual ~JsonWriter ()

  Virtual destructor for sub-classes.

Protected Member Functions

- JsonWriter (std::ostream &os, bool pretty)

  Create a new instance of the writer that will write to the specified output stream.
- std::ostream & getStream ()

  Return the output stream for use from sub-classes.
- void write (const std::string &value)

  Writes a string value to the JSON stream after escaping special characters. At present only escapes double quotes.
- void writeEmpty (const Element &element, const int32_t index)

  Serialises an empty element JSON representation.
- void writeDouble (const Element &element, const int32_t index)

  Serialises an element that holds a `double` value as JSON.
- void writeString (const Element &element, const int32_t index)

  Serialises an element that holds a `std::string` value as JSON.
- void writeDocument (const Element &element, const int32_t index)

  Writes an element that holds an embedded document as JSON.
- void writeArray (const Element &element, const int32_t index)

  Serialises an element that holds an embedded array as JSON.
- void writeBinary (const Element &element, const int32_t index)

  Serialises an element that holds embedded binary data.
- void writeOid (const Element &element, const int32_t index)

  Serialises a MongoDB OID element as JSON.
- void writeBoolean (const Element &element, const int32_t index)

  Serialises a `bool` element as JSON.
- void writeDate (const Element &element, const int32_t index)

  Serialises a date-time type element as JSON.
- void writeRegex (const Element &element, const int32_t index)

  Serialises a regular expression data type as JSON.
- void writeDbRef (const Element &element, const int32_t index)

Serialises a MongoDB database reference type element as JSON.

- void writeCode (const Element &element, const int32_t index)

  Serialises a code type element as JSON.
- void writeSymbol (const Element &element, const int32_t index)

  Serialises a programming language symbol type element as JSON.
- void writeCodeWScope (const Element &element, const int32_t index)

  Serialises a code segment with a BSON object scope as JSON.
- void writeInt (const Element &element, const int32_t index)

  Serialises a `int32_t` element as JSON.
- void writeTimestamp (const Element &element, const int32_t index)

  Serialises a MongoDB timestamp element as JSON.
- void writeLong (const Element &element, const int32_t index)

  Serialises a `int64_t` element as JSON.
- void write (const Element &element, const int32_t index)

  Serialises a element as JSON.
- void write (const Document &doc)

  Serialises a document instance as JSON.
- void write (const Array &arr)

  Serialises an array element as JSON.
- void writeIndent ()

  If pretty print is enabled, writes appropriate number of spaces to the output stream.
- void writeEndLine ()

  If pretty print is enabled, writes an end of line character to the output stream.

7.31.1 Detailed Description

A streaming writer that transforms an object model into a JSON representation.

The JSON representation is modelled upon the BSON specification and does not follow a "natural" JSON representation. This is done to make it easy to parse a JSON representation back into a uma::bson::Document or uma::bson::Array representation by the API easy.

Date

Created 2012/09/24 13:15

Copyright

Author

Rakesh

Version

Id:

JsonWriter.h 167 2012-12-13 22:11:47Z spt

7.31.2 Constructor & Destructor Documentation

7.31.2.1 virtual uma::bson::io::JsonWriter::~JsonWriter ( ) [inline], [virtual]

Virtual destructor for sub-classes.

7.31.2.2 uma::bson::io::JsonWriter::JsonWriter (std::ostream & *os,* bool *pretty*) [inline], [protected]

Create a new instance of the writer that will write to the specified output stream.

The `prettyPrint` parameter is used to indicate whether the output should contain indentation and line-breaks (primarily for human consumption) or not.

Parameters

os	The output stream to write the JSON representation to
pretty	Whether the output should contain line breaks and indentation (`true`).

7.31.3 Member Function Documentation

7.31.3.1 std::ostream& uma::bson::io::JsonWriter::getStream ( ) [inline], [protected]

Return the output stream for use from sub-classes.

Returns

std::ostream The output stream to which the JSON output is being written.

7.31.3.2 void uma::bson::io::JsonWriter::write (const std::string & *value*) [protected]

Writes a string value to the JSON stream after escaping special characters. At present only escapes double quotes.

Parameters

value	The string value to be written

7.31.3.3 void uma::bson::io::JsonWriter::write (const Element & *element,* const int32_t *index*) [protected]

Serialises a element as JSON.

Determines the data type of the element and delegates to the appropriate `writeXxx` method to serialise into JSON.

Parameters

element	The element to be serialised
index	The index of the element in the document/array.

7.31.3.4 void uma::bson::io::JsonWriter::write (const Document & *doc*) [protected]

Serialises a document instance as JSON.

Parameters

element	The document to be serialised.

7.31.3.5 void uma::bson::io::JsonWriter::write (const Array & *arr*) [protected]

Serialises an array element as JSON.

Parameters

element	The array to be serialised

7.31.3.6 void uma::bson::io::JsonWriter::writeArray (const Element & *element*, const int32_t *index*) [protected]

Serialises an element that holds an embedded array as JSON.

Writes meta data about the element, bson size and data type and delegates to write(Array) to serialise the embedded array.

Parameters

element	The element of type uma::bson::Value::Type::Array
index	The index of the element in the document/array.

7.31.3.7 void uma::bson::io::JsonWriter::writeBinary (const Element & *element*, const int32_t *index*) [protected]

Serialises an element that holds embedded binary data.

The binary data is serialised as a `base64` encoded stream of characters.

Parameters

element	The element of type uma::bson::Value::Type::BinData
index	The index of the element in the document/array.

7.31.3.8 void uma::bson::io::JsonWriter::writeBoolean (const Element & *element*, const int32_t *index*) [protected]

Serialises a `bool` element as JSON.

Parameters

element	The element of type uma::bson::Value::Type::Boolean
index	The index of the element in the document/array.

7.31.3.9 void uma::bson::io::JsonWriter::writeCode (const Element & *element*, const int32_t *index*) [protected]

Serialises a code type element as JSON.

Parameters

element	The element of type uma::bson::Value::Type::Code
index	The index of the element in the document/array.

7.31.3.10 void uma::bson::io::JsonWriter::writeCodeWScope (const Element & *element*, const int32_t *index*) [protected]

Serialises a code segment with a BSON object scope as JSON.

Parameters

element	The element of type uma::bson::Value::Type::CodeWScope
index	The index of the element in the document/array.

7.31.3.11 void uma::bson::io::JsonWriter::writeDate (const Element & *element*, const int32_t *index*) [protected]

Serialises a date-time type element as JSON.

Date-time values are serialised as an `int64_t` value that represents the milliseconds since UNIX epoch time - January 1, 1970

Parameters

element	The element of type uma::bson::Value::Type::Date
index	The index of the element in the document/array.

7.31.3.12 void uma::bson::io::JsonWriter::writeDbRef (const Element & *element*, const int32_t *index*) [protected]

Serialises a MongoDB database reference type element as JSON.

Parameters

element	The element of type uma::bson::Value::Type::DbRef
index	The index of the element in the document/array.

7.31.3.13 void uma::bson::io::JsonWriter::writeDocument (const Element & *element*, const int32_t *index*) [protected]

Writes an element that holds an embedded document as JSON.

Writes meta data about the element, bson size and data type and delegates to write(Document) to serialise the embedded document.

Parameters

element	The element of type uma::bson::Value::Type::Object
index	The index of the element in the document/array.

7.31.3.14 void uma::bson::io::JsonWriter::writeDouble (const Element & *element*, const int32_t *index*) [protected]

Serialises an element that holds a `double` value as JSON.

Parameters

element	The element of type uma::bson::Value::Type::Double
index	The index of the element in the document/array.

7.31.3.15 void uma::bson::io::JsonWriter::writeEmpty (const Element & *element*, const int32_t *index*) [protected]

Serialises an empty element JSON representation.

Empty elements are elements with any of the following value types:

- uma::bson::Value::Type::Eoo
- uma::bson::Value::Type::Undefined
- uma::bson::Value::Type::Null

Parameters

element	The element that is to be serialised as JSON
index	The index of the element in the document/array.

7.31.3.16 void uma::bson::io::JsonWriter::writeEndLine ( ) [protected]

If pretty print is enabled, writes an end of line character to the output stream.

7.31.3.17 void uma::bson::io::JsonWriter::writeIndent ( ) [protected]

If pretty print is enabled, writes appropriate number of spaces to the output stream.

7.31.3.18 void uma::bson::io::JsonWriter::writeInt (const Element & *element*, const int32_t *index*) [protected]

Serialises a `int32_t` element as JSON.

Parameters

element	The element of type uma::bson::Value::Type::Integer
index	The index of the element in the document/array.

7.31.3.19 void uma::bson::io::JsonWriter::writeLong (const Element & *element*, const int32_t *index*) [protected]

Serialises a `int64_t` element as JSON.

Parameters

element	The element of type uma::bson::Value::Type::Long
index	The index of the element in the document/array.

7.31.3.20 void uma::bson::io::JsonWriter::writeOid (const Element & *element*, const int32_t *index*) [protected]

Serialises a MongoDB OID element as JSON.

Serialises a 24 character string representation of the object Id.

Parameters

element	The element of type uma::bson::Value::Type::OID
index	The index of the element in the document/array.

7.31.3.21 void uma::bson::io::JsonWriter::writeRegex (const Element & *element*, const int32_t *index*) [protected]

Serialises a regular expression data type as JSON.

Parameters

element	The element of type uma::bson::Value::Type::RegEx
index	The index of the element in the document/array.

7.31.3.22 void uma::bson::io::JsonWriter::writeString (const Element & *element*, const int32_t *index*) [protected]

Serialises an element that holds a `std::string` value as JSON.

Parameters

element	The element of type uma::bson::Value::Type::String
index	The index of the element in the document/array.

7.31.3.23 void uma::bson::io::JsonWriter::writeSymbol (const Element & *element*, const int32_t *index*) [protected]

Serialises a programming language symbol type element as JSON.

Parameters

element	The element of type uma::bson::Value::Type::Symbol
index	The index of the element in the document/array.

7.31.3.24 void uma::bson::io::JsonWriter::writeTimestamp (const Element & *element*, const int32_t *index*) [protected]

Serialises a MongoDB timestamp element as JSON.

Timestamp elements are represented by a `int32_t` increment value and a `std::time_t` type date value.

Parameters

element	The element of type uma::bson::Value::Type::Timestamp
index	The index of the element in the document/array.

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

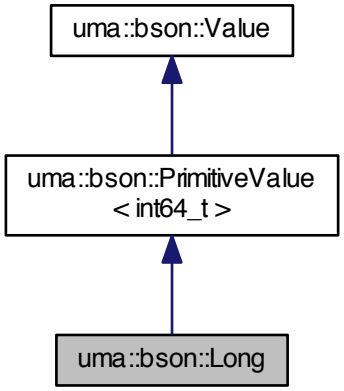
- /Users/rakesh/projects/opensource/mongoviewer/bson/src/api/uma/bson/io/JsonWriter.h

7.32 uma::bson::Long Class Reference

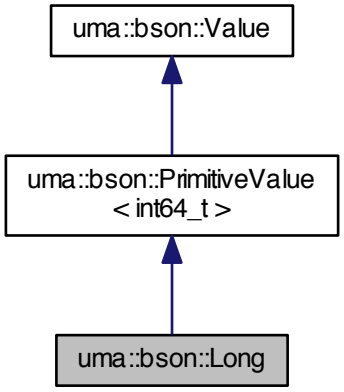
A POD that represents a long (64 bit integer) type BSON element value.

`#include <Long.h>`

Inheritance diagram for uma::bson::Long:



Collaboration diagram for uma::bson::Long:



Public Member Functions

- Long ()

  Default CTOR.
- Long (const int64_t v)

  Create a new instance that encapsulates the specified value.
- int32_t getSize () const

  Return the size in bytes of the BSON representation of this value.
- Value::Type getType () const

  Returns the type for this instance as listed in the BSON specifications.

Additional Inherited Members

7.32.1 Detailed Description

A POD that represents a long (64 bit integer) type BSON element value.

Date

Created 2012/09/13 18:17

Copyright

Author

Rakesh

Version

Id:

Long.h 172 2012-12-14 17:21:21Z spt

7.32.2 Constructor & Destructor Documentation

7.32.2.1 uma::bson::Long::Long ( ) [inline]

Default CTOR.

7.32.2.2 uma::bson::Long::Long (const int64_t *v*) [inline]

Create a new instance that encapsulates the specified value.

Parameters

v	The value to encapsulate

7.32.3 Member Function Documentation

7.32.3.1 int32_t uma::bson::Long::getSize ( ) const [inline], [virtual]

Return the size in bytes of the BSON representation of this value.

Returns

Returns 8 as per specifications

Implements uma::bson::Value.

7.32.3.2 Value::Type uma::bson::Long::getType ( ) const [inline], [virtual]

Returns the type for this instance as listed in the BSON specifications.

Returns

Returns uma::bson::Value::Type::Long

Implements uma::bson::Value.

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

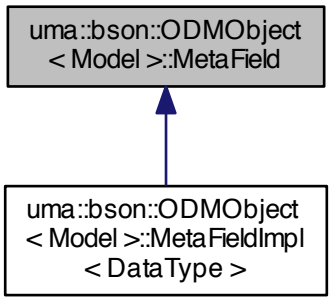
- /Users/rakesh/projects/opensource/mongoviewer/bson/src/api/uma/bson/Long.h

7.33 uma::bson::ODMObject< Model >::MetaField Class Reference

Abstract base class that encapsulates a field in a model object. Primarily used to get around requirement that implementation needs exact type of data encapsulated in the field as template type.

`#include <ODMObject.h>`

Inheritance diagram for uma::bson::ODMObject< Model >::MetaField:



Public Member Functions

- MetaField (const std::string &field)
  Create a new instance representing the field with the specified name.
- virtual ~MetaField ()
  Virtual dtor for sub-classing.
- virtual Value & getValue (Model *model)=0
  Return the value for the field represented by this instance.
- virtual void setValue (Model *model, const Value &value)=0
  Set the value for the field represented by this instance.
- const std::string & getName () const
  Return the name of the field encapsulated in this meta field instance.

7.33.1 Detailed Description

template<typename Model>class uma::bson::ODMObject< Model >::MetaField

Abstract base class that encapsulates a field in a model object. Primarily used to get around requirement that implementation needs exact type of data encapsulated in the field as template type.

Template Parameters

Model	The type of the class whose field is to be represented

7.33.2 Constructor & Destructor Documentation

7.33.2.1 template<typename Model > uma::bson::ODMObject< Model >::MetaField::MetaField (const std::string & *field*) [inline]

Create a new instance representing the field with the specified name.

Parameters

field	The name of the field.

7.33.2.2 template<typename Model > virtual uma::bson::ODMObject< Model >::MetaField::~MetaField ( ) [inline], [virtual]

Virtual dtor for sub-classing.

7.33.3 Member Function Documentation

7.33.3.1 template<typename Model > const std::string& uma::bson::ODMObject< Model >::MetaField::getName ( ) const [inline]

Return the name of the field encapsulated in this meta field instance.

Returns

const std::string The field name.

7.33.3.2 template<typename Model > virtual Value& uma::bson::ODMObject< Model >::MetaField::getValue (Model * *model*) [pure virtual]

Return the value for the field represented by this instance.

Parameters

model	The model object from which the value is to retrieved.

Returns

Value The value for the field.

Implemented in uma::bson::ODMObject< Model >::MetaFieldImpl< DataType >.

7.33.3.3 template<typename Model > virtual void uma::bson::ODMObject< Model >::MetaField::setValue (Model * *model*, const Value & *value*) [pure virtual]

Set the value for the field represented by this instance.

Parameters

model	The model object in which the value is to be stored.
value	The value for the field to store.

Implemented in uma::bson::ODMObject< Model >::MetaFieldImpl< DataType >.

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

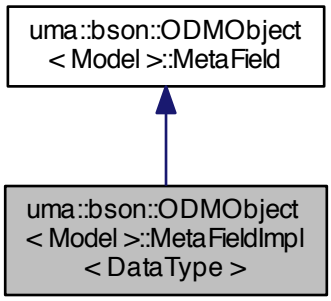
- /Users/rakesh/projects/opensource/mongoviewer/bson/src/api/uma/bson/ODMObject.h

7.34 uma::bson::ODMObject< Model >::MetaFieldImpl< DataType > Class Template Reference

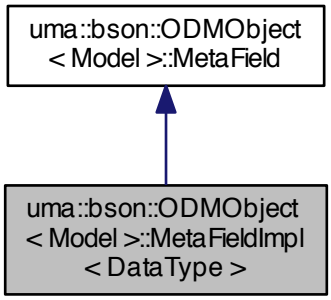
Encapsulates a field in a model object. Fields are represented as a triplet of the field in the class, and its accessor and mutator methods.

`#include <ODMObject.h>`

Inheritance diagram for uma::bson::ODMObject< Model >::MetaFieldImpl< DataType >:



Collaboration diagram for uma::bson::ODMObject< Model >::MetaFieldImpl< DataType >:



Public Types

- typedef DataType &(Model::* Accessor)()

Member function pointer to the accessor method.

- typedef void(Model::* Mutator)(const DataType &)

  Member function pointer to the mutator method.

Public Member Functions

- MetaFieldImpl (const std::string &field, Accessor get, Mutator set)

  Create a new field instance for the specified instance and methods.
- Value & getValue (Model *model)

  Return the value for the field represented by this instance.
- void setValue (Model *model, const Value &value)

  Set the value for the field represented by this instance.

7.34.1 Detailed Description

template<typename Model>template<typename DataType>class uma::bson::ODMObject< Model >::MetaFieldImpl< DataType >

Encapsulates a field in a model object. Fields are represented as a triplet of the field in the class, and its accessor and mutator methods.

Template Parameters

Model	The type of the class whose field is to be represented. This is implied by the template parameter for the wrapper ODMObject class.
DataType	The data type for the field. Must be a sub-class of uma::bson::Value.

7.34.2 Member Typedef Documentation

7.34.2.1 template<typename Model > template<typename DataType > typedef DataType&(Model::* uma::bson::ODMObject< Model >::MetaFieldImpl< DataType >::Accessor)()

Member function pointer to the accessor method.

7.34.2.2 template<typename Model > template<typename DataType > typedef void(Model::* uma::bson::ODMObject< Model >::MetaFieldImpl< DataType >::Mutator)(const DataType &)

Member function pointer to the mutator method.

7.34.3 Constructor & Destructor Documentation

7.34.3.1 template<typename Model > template<typename DataType > uma::bson::ODMObject< Model >::MetaFieldImpl< DataType >::MetaFieldImpl (const std::string & *field*, Accessor *get*, Mutator *set*) [inline]

Create a new field instance for the specified instance and methods.

Parameters

field	The name of the field.
get	The accessor method for the field.
set	The mutator method for the field.

7.34.4 Member Function Documentation

7.34.4.1 template<typename Model > template<typename DataType > Value& uma::bson::ODMObject< Model >::MetaFieldImpl< DataType >::getValue (Model * *model*) [inline], [virtual]

Return the value for the field represented by this instance.

Parameters

model	The model object from which the value is to retrieved.

Returns

Value The value for the field.

Implements uma::bson::ODMObject< Model >::MetaField.

7.34.4.2 template<typename Model > template<typename DataType > void uma::bson::ODMObject< Model >::MetaFieldImpl< DataType >::setValue (Model * *model*, const Value & *value*) [inline], [virtual]

Set the value for the field represented by this instance.

Parameters

model	The model object in which the value is to be stored.
value	The value for the field to store.

Implements uma::bson::ODMObject< Model >::MetaField.

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

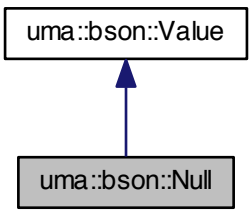
- /Users/rakesh/projects/opensource/mongoviewer/bson/src/api/uma/bson/ODMObject.h

7.35 uma::bson::Null Class Reference

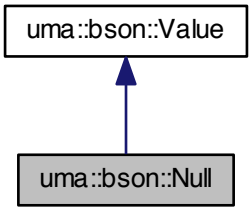
A POD that represents a null BSON element value.

`#include <Null.h>`

Inheritance diagram for uma::bson::Null:



Collaboration diagram for uma::bson::Null:



Public Member Functions

- Null ()
  Default CTOR.
- int32_t getSize () const
  Returns the size of the object as per BSON specifications.
- Value::Type getType () const
  Returns the type for this instance as listed in the BSON specifications.

Additional Inherited Members

7.35.1 Detailed Description

A POD that represents a null BSON element value.

Date

Created 2012/09/09 20:16

Copyright

Author

Rakesh

Version

Id:

Null.h 172 2012-12-14 17:21:21Z spt

7.35.2 Constructor & Destructor Documentation

7.35.2.1 uma::bson::Null::Null ( ) [inline]

Default CTOR.

7.35.3 Member Function Documentation

7.35.3.1 int32_t uma::bson::Null::getSize ( ) const [inline], [virtual]

Returns the size of the object as per BSON specifications.

Returns

Returns 0

Implements uma::bson::Value.

7.35.3.2 Value::Type uma::bson::Null::getType ( ) const [inline], [virtual]

Returns the type for this instance as listed in the BSON specifications.

Returns

Returns uma::bson::Value::Type::Null

Implements uma::bson::Value.

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

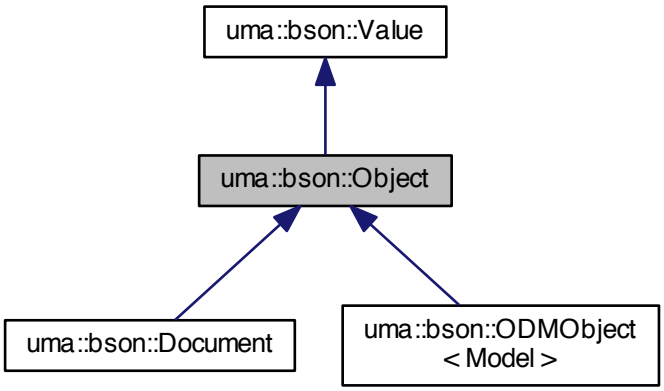
- /Users/rakesh/projects/opensource/mongoviewer/bson/src/api/uma/bson/Null.h

7.36 uma::bson::Object Class Reference

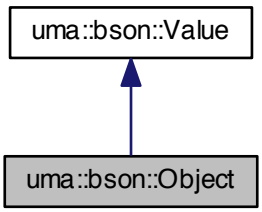
Abstract base class that represents a BSON Object type.

`#include <Object.h>`

Inheritance diagram for uma::bson::Object:



Collaboration diagram for uma::bson::Object:



Public Types

- typedef std::vector< std::string > FieldNames

  A container for the top-level fields in an object.

- typedef FieldNames::const_iterator FieldsIterator

  Iterator for iterating over the top-level field names in an object.

Public Member Functions

- virtual ~Object ()

  Virtual destructor for sub-classes.

- virtual const FieldNames getFieldNames () const =0

  Return a vector of all the element names (top-level only) in this object.

- virtual Value & getValue (const std::string &name)=0

  Return the value for the element with the specified name.

- virtual const Value & getValue (const std::string &name) const

  Constant version of getValue method.

- virtual Object * getObjectForArray (const std::string &)

  Return an empty instance of an object implementation that represents the type of data stored in an array type field.

- virtual void setValue (const std::string &name, const Value &value)

  Set the value for the element with the specified name.

- virtual void toBson (std::ostream &os) const

  Serialise the data in this object in BSON format to the specified output stream.

- virtual void populate (std::istream &is)

  Populate the fields in this object from the BSON data stream. Complementary method to toBson.

- int32_t getSize () const

  Return the size of the data as per BSON specifications.

- Value::Type getType () const

  Returns the type for this instance as listed in the BSON specifications.

Additional Inherited Members

7.36.1 Detailed Description

Abstract base class that represents a BSON Object type.

The primary purpose of this indirection is to allow client API to model BSON object instances as regular POD types. By implementing the methods in this abstract interface these classes automatically get support for serialisation to and from BSON.

Since

Version 2.2

Date

Created 2012/12/17 19:30

Author

Rakesh

Version

Id:

Object.h 192 2012-12-23 22:51:30Z spt

7.36.2 Member Typedef Documentation

7.36.2.1 typedef std::vector<std::string> uma::bson::Object::FieldNames

A container for the top-level fields in an object.

7.36.2.2 typedef FieldNames::const_iterator uma::bson::Object::FieldsIterator

Iterator for iterating over the top-level field names in an object.

7.36.3 Constructor & Destructor Documentation

7.36.3.1 virtual uma::bson::Object::~Object ( ) [inline], [virtual]

Virtual destructor for sub-classes.

7.36.4 Member Function Documentation

7.36.4.1 virtual const FieldNames uma::bson::Object::getFieldNames ( ) const [pure virtual]

Return a vector of all the element names (top-level only) in this object.

Returns

The vector of element names.

Implemented in uma::bson::Document, and uma::bson::ODMObject< Model >.

7.36.4.2 virtual Object* uma::bson::Object::getObjectForArray (const std::string &) [inline], [virtual]

Return an empty instance of an object implementation that represents the type of data stored in an array type field.

Array type fields are used to represent data stored in container classes in the class. This method is used primarily when de-serialising an object instance from its BSON representation. If the object being de-serialised contains array type data, this method will be used to determine the type of values stored in the array.

Note that container fields that store simple types (literally all types of values except Value::Type::Object) will in general not need any special support or this method to be implemented.

Note that the default implementation just throws an exception. This is done to avoid having to make this method a pure-virtual method. Not all model objects will need to store collections of values, and hence only objects that need to support collections need implement this method.

The calling method implemented in this API will populate the instance returned with the BSON data and save in an uma::bson::Element. The element instance takes ownership of the memory allocated for the instance returned. Implementations must not `delete` the instance returned.

Parameters

name	The name of the array type field.

Returns

A new empty object instance that will be populated and stored in an uma::bson::Array.

7.36.4.3 int32_t uma::bson::Object::getSize ( ) const [virtual]

Return the size of the data as per BSON specifications.

Returns

The size of the BSON data.

Implements uma::bson::Value.

7.36.4.4 Value::Type uma::bson::Object::getType ( ) const [inline], [virtual]

Returns the type for this instance as listed in the BSON specifications.

Returns

Returns uma::bson::Value::Type::Object

Implements uma::bson::Value.

7.36.4.5 virtual Value& uma::bson::Object::getValue (const std::string & *name*) [pure virtual]

Return the value for the element with the specified name.

This method is primarily intended to provide rudimentary ODM (object-document mapping) capabilities in the API. Model objects that extend this class and store fields as Value types may be serialised to BSON seamlessly. See unit test suite for a simple custom model object that is serialised to BSON.

Since

Version 2.2

Parameters

name	The name of the BSON element

Returns

The value for the element.

Exceptions

Poco::NotFoundException	If no element with specified name exists.

Implemented in uma::bson::Document, and uma::bson::ODMObject< Model >.

7.36.4.6 virtual const Value& uma::bson::Object::getValue (const std::string & *name*) const [inline], [virtual]

Constant version of getValue method.

7.36.4.7 virtual void uma::bson::Object::populate (std::istream & *is*) [virtual]

Populate the fields in this object from the BSON data stream. Complementary method to toBson.

Parameters

is	The BSON data stream from which this object instance is to be populated.

Exceptions

Poco::InvalidArgument-Exception	If the bson data contains elements that do not match the fields in this class.

7.36.4.8 virtual void uma::bson::Object::setValue (const std::string & *name*, const Value & *value*) [virtual]

Set the value for the element with the specified name.

This method is also used to provide rudimentary ODM capabilities in the API. Model objects that extend this class and store fields as Value types may be deserialised from BSON through this method. See unit test suite for a simple custom model object that is de-serialised from BSON.

Note that from Version 2.3 onwards a default implementation is provided. The default implementation handles setting the value for all the simple types. In particular `Value::Type::Object`, `Value::Type::Array` and consequently `Value::Type::CodeWScope` are not handled and may throw an exception. Despite this clients may still not need to provide an implementation of this method even if they store other object and array fields. The BSON deserialiser implemented in this API retrieves a reference to the current object/array from the destination instance and populates those. Hence if the only code using this method is from the internal BSON deserialiser, there will be no need to provide additional implementation for setting object/array fields.

Since

Version 2.2

Parameters

name	
value	

Exceptions

Poco::InvalidArgument-Exception	Will be thrown if the specified value type does not match the existing value type. May also be throw since this method invokes getValue(const std::string&) at the beginning to fetch the current instance that is to be modified.
Poco::NotImplemented-Exception	If the value specified is a complex type such as `Value::Type::Object` or `Value::Type::Array`.

Reimplemented in uma::bson::Document, and uma::bson::ODMObject< Model >.

7.36.4.9 virtual void uma::bson::Object::toBson (std::ostream & *os*) const [virtual]

Serialise the data in this object in BSON format to the specified output stream.

The default implementation uses uma::bson::io::ObjectWriter to serialise the fields returned by getFieldNames to the stream.

Parameters

os	The output stream to serialise the BSON data to.

Reimplemented in uma::bson::Document.

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

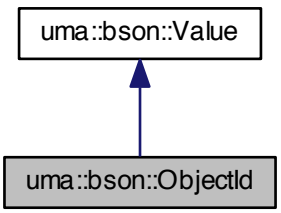
- /Users/rakesh/projects/opensource/mongoviewer/bson/src/api/uma/bson/Object.h

7.37 uma::bson::ObjectId Class Reference

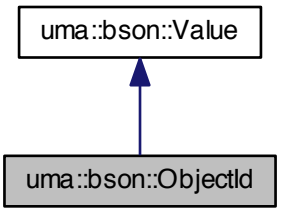
A POD that represents a MongoDB OID type.

`#include <ObjectId.h>`

Inheritance diagram for uma::bson::ObjectId:



Collaboration diagram for uma::bson::ObjectId:



Public Member Functions

- ObjectId (const Poco::Timestamp &ts=Poco::Timestamp())

  Default constructor. Create a new instance with the the current timestamp or specified value.
- ObjectId (const char(&value)[12])

  Create a new OID from the byte array specified.
- ObjectId (const std::string &str)

  Create oid from 24 char hex string.
- ObjectId (const Poco::Timestamp &date, const int random, const int sequence)

  Create a new OID with the three components that the OID bytes represent.
- const Poco::Timestamp getTime () const

  Return the timestamp value encoded in this OID.
- const std::string toString () const

  Return a 24 char hex representation of the OID.
- void getBytes (char ∗bytes, const int max=12) const

  Populate the specified char array with the contents of the OID.
- void setBytes (const char(&bytes)[12])

  Replaces the bytes represented in this instance with the specified byte array.
- int32_t getSize () const

  Return the size in bytes of the BSON representation of this OID value.
- Value::Type getType () const

  Returns the type for this instance as listed in the BSON specifications.

Additional Inherited Members

7.37.1 Detailed Description

A POD that represents a MongoDB OID type.

A POD that represents a MongoDB OID type. OID's encode the creation time of the OID, and hence may be used as an efficient creation date property for the document.

Date

Created 2012/09/06 19:04

Copyright

Author

Rakesh

Version

Id:

ObjectId.h 192 2012-12-23 22:51:30Z spt

7.37.2 Constructor & Destructor Documentation

7.37.2.1 uma::bson::ObjectId::ObjectId (const Poco::Timestamp & *ts* = `Poco::Timestamp()`) `[inline]`, `[explicit]`

Default constructor. Create a new instance with the the current timestamp or specified value.

Note that this version stores the microsecond time part in the last 4 bytes of the OID. This allows a very precise timestamp for the OID value to be retrieved. The middle 4 bytes are still populated using a random sequence, which hopefully will make the generated OID value close to globally unique.

Parameters

ts	The timestamp to initialise the OID from. Defaults to current time.

7.37.2.2 uma::bson::ObjectId::ObjectId (const char(&) *value[12]*) `[inline]`, `[explicit]`

Create a new OID from the byte array specified.

Parameters

(value)[]	The 12 byte array to initialise from

7.37.2.3 uma::bson::ObjectId::ObjectId (const std::string & *str*) `[inline]`, `[explicit]`

Create oid from 24 char hex string.

Parameters

str	The string representation from which to initialise the OID

7.37.2.4 uma::bson::ObjectId::ObjectId (const Poco::Timestamp & *date*, const int *random*, const int *sequence*) `[inline]`

Create a new OID with the three components that the OID bytes represent.

Parameters

date	The time stamp used to initialise the instance. Note that only the seconds from UNIX epoch will be used to populate the first 4 bytes
random	The random value used to populate the second 4 bytes.
sequence	The sequence number used to populate the last 4 bytes.

7.37.3 Member Function Documentation

7.37.3.1 void uma::bson::ObjectId::getBytes (char * *bytes*, const int *max* = 12) const [inline]

Populate the specified char array with the contents of the OID.

Parameters

bytes	The byte array (ideally 12 char in length) that is to be populated with the contents of the OID value.
max	The optional maximum number of characters to write into the `bytes` array. Defaults to `12`

7.37.3.2 int32_t uma::bson::ObjectId::getSize ( ) const [inline], [virtual]

Return the size in bytes of the BSON representation of this OID value.

Returns

Return `12` as per specifications

Implements uma::bson::Value.

7.37.3.3 const Poco::Timestamp uma::bson::ObjectId::getTime ( ) const [inline]

Return the timestamp value encoded in this OID.

Returns

The timestamp value.

7.37.3.4 Value::Type uma::bson::ObjectId::getType ( ) const [inline], [virtual]

Returns the type for this instance as listed in the BSON specifications.

Returns

Returns uma::bson::Value::Type::OID

Implements uma::bson::Value.

7.37.3.5 void uma::bson::ObjectId::setBytes (const char(&) *bytes[12]*) [inline]

Replaces the bytes represented in this instance with the specified byte array.

Since

Version 2.3

Parameters

The	12 byte char array to assign to this object id.

7.37.3.6 const std::string uma::bson::ObjectId::toString ( ) const [inline]

Return a 24 char hex representation of the OID.

Returns

The hex representation

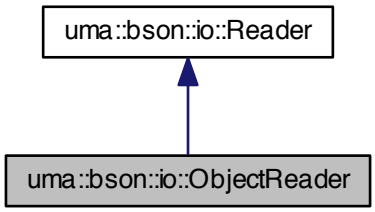
The documentation for this class was generated from the following file:

- /Users/rakesh/projects/opensource/mongoviewer/bson/src/api/uma/bson/ObjectId.h

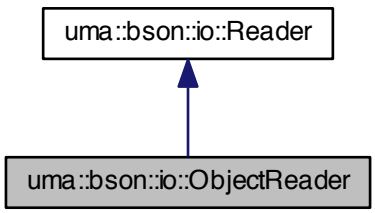
7.38 uma::bson::io::ObjectReader Class Reference

`#include <ObjectReader.h>`

Inheritance diagram for uma::bson::io::ObjectReader:



Collaboration diagram for uma::bson::io::ObjectReader:



Public Member Functions

- virtual void parse (uma::bson::Object &object, const std::string &filePath)

  Parses the contents of the file at the path specified into the specified object instance.
- virtual void parse (uma::bson::Object &object, std::istream &fin)

  Parses the BSON data bytes from the specified input stream into the specified object instance.

Protected Member Functions

- void readObject (uma::bson::Object &object, std::istream &is)
- void readArray (uma::bson::Object &object, const std::string &name, uma::bson::Array &array, std::istream &fin)

7.38.1 Member Function Documentation

7.38.1.1 virtual void uma::bson::io::ObjectReader::parse (uma::bson::Object & *object,* const std::string & *filePath*) [virtual]

Parses the contents of the file at the path specified into the specified object instance.

Parameters

object	The object instance whose fields are to be populated with the BSON data from the specified file.
filePath	The fully qualified path to the BSON file.

7.38.1.2 virtual void uma::bson::io::ObjectReader::parse (uma::bson::Object & *object,* std::istream & *fin*) [virtual]

Parses the BSON data bytes from the specified input stream into the specified object instance.

Parameters

object	The object instance whose fields are to be populated with the BSON data from the specified file.
fin	The input stream (file or network) from which the BSON data is to be read.

7.38.1.3 void uma::bson::io::ObjectReader::readArray (uma::bson::Object & *object,* const std::string & *name,* uma::bson::Array & *array,* std::istream & *fin*) [protected]

7.38.1.4 void uma::bson::io::ObjectReader::readObject (uma::bson::Object & *object,* std::istream & *is*) [protected]

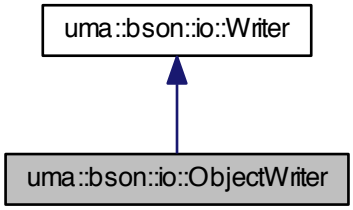
The documentation for this class was generated from the following file:

- /Users/rakesh/projects/opensource/mongoviewer/bson/src/api/uma/bson/io/ObjectReader.h

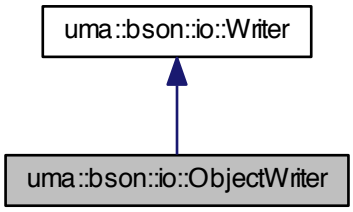
7.39 uma::bson::io::ObjectWriter Class Reference

```
#include <ObjectWriter.h>
```

Inheritance diagram for uma::bson::io::ObjectWriter:



Collaboration diagram for uma::bson::io::ObjectWriter:



Public Member Functions

- ObjectWriter (const uma::bson::Object &obj, std::ostream &os)

  Create a new instance of the writer to serialise the specified object to the output stream.
- virtual ~ObjectWriter ()

  Virtual destructor for sub-classes.
- virtual void write ()

  Serialise the object instance to the output stream. Sub-classes may re-implement to customise serialisation of custom model objects.

Protected Member Functions

- const uma::bson::Object & getObject () const

  Return the object that is being serialised using this instance.
- virtual void write (const Object &obj)

  Serialises a BSON object implementation to the output stream. This method is used primarily for serialising sub-classes of uma::bson::Object which implement simple object models. Default uma::bson::Document instances are serialised using the more specialised write(const uma::bson::Document&) method.
- void write (const Array &arr)

  Serialises an array to the output stream.

7.39.1 Detailed Description

A base streaming writer class that transforms the data in a bson object into a BSON representation and writes it to an output stream.

Since

Version 2.2

Date

Created 2012/12/18 07:25

Copyright

Author

Rakesh

Version

Id:

ObjectWriter.h 181 2012-12-21 01:08:58Z spt

7.39.2 Constructor & Destructor Documentation

7.39.2.1 uma::bson::io::ObjectWriter::ObjectWriter (const uma::bson::Object & *obj*, std::ostream & *os*)

Create a new instance of the writer to serialise the specified object to the output stream.

Parameters

obj	The object to serialise
os	The output stream to write to.

7.39.2.2 virtual uma::bson::io::ObjectWriter::~ObjectWriter ( ) [inline], [virtual]

Virtual destructor for sub-classes.

7.39.3 Member Function Documentation

7.39.3.1 const uma::bson::Object& uma::bson::io::ObjectWriter::getObject ( ) const [inline], [protected]

Return the object that is being serialised using this instance.

Returns

The object that is being serialised using this writer.

7.39.3.2 virtual void uma::bson::io::ObjectWriter::write ( ) [virtual]

Serialise the object instance to the output stream. Sub-classes may re-implement to customise serialisation of custom model objects.

7.39.3.3 virtual void uma::bson::io::ObjectWriter::write (const Object & *obj*) [protected], [virtual]

Serialises a BSON object implementation to the output stream. This method is used primarily for serialising sub-classes of uma::bson::Object which implement simple object models. Default uma::bson::Document instances are serialised using the more specialised write(const uma::bson::Document&) method.

Parameters

obj	The object to serialise to BSON.

7.39.3.4 void uma::bson::io::ObjectWriter::write (const Array & *arr*) [protected], [virtual]

Serialises an array to the output stream.

Parameters

value	The value to serialise to BSON

Reimplemented from uma::bson::io::Writer.

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

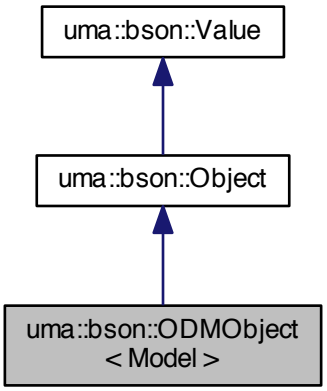
- /Users/rakesh/projects/opensource/mongoviewer/bson/src/api/uma/bson/io/ObjectWriter.h

7.40 uma::bson::ODMObject< Model > Class Template Reference

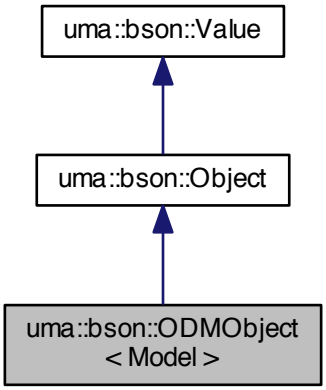
Abstract class that presents a more friendly ODM interface than uma::bson::Object.

`#include <ODMObject.h>`

Inheritance diagram for uma::bson::ODMObject< Model >:



Collaboration diagram for uma::bson::ODMObject< Model >:



Classes

- class MetaField

  Abstract base class that encapsulates a field in a model object. Primarily used to get around requirement that implementation needs exact type of data encapsulated in the field as template type.
- class MetaFieldImpl

  Encapsulates a field in a model object. Fields are represented as a triplet of the field in the class, and its accessor and mutator methods.

Public Member Functions

- virtual ~ODMObject ()

  Virtual destructor for sub-classes.
- const FieldNames getFieldNames () const

  Return the names of the fields registered. Note that the fields are stored in a `std::map`*, hence the order of fields returned may not correspond to the order in which they were registered.*
- Value & getValue (const std::string &name)

  Return the value for the element with the specified name.
- void setValue (const std::string &name, const Value &value)

  Set the value for the element with the specified name.

Protected Types

- typedef Poco::SharedPtr
  < MetaField > MetaFieldPtr

  Shared pointer for a meta field instance.

Protected Member Functions

- void registerField (const MetaFieldPtr mo)

Add (register) a field with the map of fields in the model object. Best practise is to check whether class has been registered (see registered, and if not, register all the fields for the class.

- MetaFieldPtr getMetaField (const std::string &name)

  Return the meta field instance for the field with the specified name.

Static Protected Member Functions

- static bool registered (Model *model)

  Check to see if the class for the model object has been registered. Registration is the process of defining the fields with their accessor and mutator methods.

Additional Inherited Members

7.40.1 Detailed Description

template<typename Model>class uma::bson::ODMObject< Model >

Abstract class that presents a more friendly ODM interface than uma::bson::Object.

This class uses a registration system for model objects to register their serialisable fields along with their accessor and mutator methods (see registerField. This removes the requirement to implement getValue using an ugly `if-else` statement block.

Warning

Note: Model classes will need to provide a `non-const` version of the accessor methods for the registration to work.

Template Parameters

Model	The type of the sub-class (needed for the MetaField implementation).

Since

Version 2.5

Date

Created 2013/02/13 10:30

Author

Rakesh

Version

Id:

ODMObject.h 207 2013-02-28 16:03:52Z spt

7.40.2 Member Typedef Documentation

7.40.2.1 template<typename Model > typedef Poco::SharedPtr<MetaField> uma::bson::ODMObject< Model >::MetaFieldPtr [protected]

Shared pointer for a meta field instance.

7.40.3 Constructor & Destructor Documentation

7.40.3.1 template<typename Model > virtual uma::bson::ODMObject< Model >::~ODMObject ( ) [inline], [virtual]

Virtual destructor for sub-classes.

7.40.4 Member Function Documentation

7.40.4.1 template<typename Model > const FieldNames uma::bson::ODMObject< Model >::getFieldNames ( ) const [inline], [virtual]

Return the names of the fields registered. Note that the fields are stored in a `std::map`, hence the order of fields returned may not correspond to the order in which they were registered.

Returns

const FieldNames The `vector` of field names.

Implements uma::bson::Object.

7.40.4.2 template<typename Model > MetaFieldPtr uma::bson::ODMObject< Model >::getMetaField (const std::string & *name*) [inline], [protected]

Return the meta field instance for the field with the specified name.

Parameters

name	The name of the field whose meta instance is to be retrieved.

Returns

MetaFieldPtr The meta field representing the named field.

7.40.4.3 template<typename Model > Value& uma::bson::ODMObject< Model >::getValue (const std::string & *name*) [inline], [virtual]

Return the value for the element with the specified name.

This method is primarily intended to provide rudimentary ODM (object-document mapping) capabilities in the API. Model objects that extend this class and store fields as Value types may be serialised to BSON seamlessly. See unit test suite for a simple custom model object that is serialised to BSON.

Since

Version 2.2

Parameters

name	The name of the BSON element

Returns

The value for the element.

Exceptions

Poco::NotFoundException	If no element with specified name exists.

Implements uma::bson::Object.

7.40.4.4 template<typename Model > static bool uma::bson::ODMObject< Model >::registered (Model * *model*) [inline], [static], [protected]

Check to see if the class for the model object has been registered. Registration is the process of defining the fields with their accessor and mutator methods.

Parameters

model	The model object to check to see if registered.

Returns

bool Returns `true` if already registered.

7.40.4.5 template<typename Model > void uma::bson::ODMObject< Model >::registerField (const MetaFieldPtr *mo*) [inline], [protected]

Add (register) a field with the map of fields in the model object. Best practise is to check whether class has been registered (see registered, and if not, register all the fields for the class.

Parameters

mo	The shared pointer to the meta field instance.

7.40.4.6 template<typename Model > void uma::bson::ODMObject< Model >::setValue (const std::string & *name*, const Value & *value*) [inline], [virtual]

Set the value for the element with the specified name.

This method is also used to provide rudimentary ODM capabilities in the API. Model objects that extend this class and store fields as Value types may be deserialised from BSON through this method. See unit test suite for a simple custom model object that is de-serialised from BSON.

Note that from Version 2.3 onwards a default implementation is provided. The default implementation handles setting the value for all the simple types. In particular `Value::Type::Object`, `Value::Type::Array` and consequently `Value::Type::CodeWScope` are not handled and may throw an exception. Despite this clients may still not need to provide an implementation of this method even if they store other object and array fields. The BSON deserialiser implemented in this API retrieves a reference to the current object/array from the destination instance and populates those. Hence if the only code using this method is from the internal BSON deserialiser, there will be no need to provide additional implementation for setting object/array fields.

Since

Version 2.2

Parameters

name	
value	

Exceptions

Poco::InvalidArgument-Exception	Will be thrown if the specified value type does not match the existing value type. May also be throw since this method invokes getValue(const std::string&) at the beginning to fetch the current instance that is to be modified.
Poco::NotImplemented-Exception	If the value specified is a complex type such as `Value::Type::Object` or `Value::Type::Array`.

Reimplemented from uma::bson::Object.

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

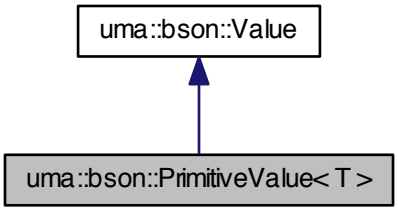
- /Users/rakesh/projects/opensource/mongoviewer/bson/src/api/uma/bson/ODMObject.h

7.41 uma::bson::PrimitiveValue< T > Class Template Reference

A base value class used to represent primitive type values.

`#include <PrimitiveValue.h>`

Inheritance diagram for uma::bson::PrimitiveValue< T >:



Collaboration diagram for uma::bson::PrimitiveValue< T >:

Public Member Functions

- PrimitiveValue ()

Default CTOR.

- PrimitiveValue (const T v)

  Create a new instance that encapsulates the specified value.

- T getValue () const

  Return the primitive value encapsulated in this instance.

- PrimitiveValue< T > & setValue (const T v)

  Set the value encapsulated in this instance.

- PrimitiveValue< T > & operator+= (const T v)

  Adds the specified value to the encapsulated value.

- PrimitiveValue< T > & operator-= (const T v)

  Subtract the specified value from the encapsulated value.

- PrimitiveValue< T > & operator*= (const T v)

  Multiple the encapsulated value by the specified value.

- PrimitiveValue< T > & operator/= (const T v)

  Divide the encapsulated value by the specified value.

- operator T () const

  Cast operator to convert to the encapsulated value.

Additional Inherited Members

7.41.1 Detailed Description

template<typename T>class uma::bson::PrimitiveValue< T >

A base value class used to represent primitive type values.

Template Parameters

T	The primitive type encapsulated by this class.

Since

Version 2.0

Date

Created 2012/12/02 18:02

Copyright

Author

Rakesh

Version

Id:

PrimitiveValue.h 203 2013-02-15 12:23:15Z spt

7.41.2 Constructor & Destructor Documentation

7.41.2.1 template<typename T> uma::bson::PrimitiveValue< T >::PrimitiveValue ( ) [inline]

Default CTOR.

7.41.2.2 template<typename T> uma::bson::PrimitiveValue< T >::PrimitiveValue (const T *v*) [inline], [explicit]

Create a new instance that encapsulates the specified value.

Parameters

v	The primitive value to encapsulate

7.41.3 Member Function Documentation

7.41.3.1 template<typename T> T uma::bson::PrimitiveValue< T >::getValue ( ) const [inline]

Return the primitive value encapsulated in this instance.

Returns

The primitive value

7.41.3.2 template<typename T> uma::bson::PrimitiveValue< T >::operator T ( ) const [inline]

Cast operator to convert to the encapsulated value.

7.41.3.3 template<typename T> PrimitiveValue<T>& uma::bson::PrimitiveValue< T >::operator*= (const T *v*) [inline]

Multiple the encapsulated value by the specified value.

Parameters

v	The value to multiple by

Returns

This instance for method chaining

7.41.3.4 template<typename T> PrimitiveValue<T>& uma::bson::PrimitiveValue< T >::operator+= (const T *v*) [inline]

Adds the specified value to the encapsulated value.

Parameters

v	The value to add

Returns

This instance for method chaining

7.41.3.5 template<typename T> PrimitiveValue<T>& uma::bson::PrimitiveValue< T >::operator-= (const T *v*) [inline]

Subtract the specified value from the encapsulated value.

Parameters

v	The value to subtract

Returns

This instance for method chaining

7.41.3.6 template<typename T> PrimitiveValue<T>& uma::bson::PrimitiveValue< T >::operator/= (const T *v*) [inline]

Divide the encapsulated value by the specified value.

Parameters

v	The value to divide by

Returns

This instance for method chaining

7.41.3.7 template<typename T> PrimitiveValue<T>& uma::bson::PrimitiveValue< T >::setValue (const T *v*) [inline]

Set the value encapsulated in this instance.

Parameters

v	The new value to set.

Returns

This instance for method chaining

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

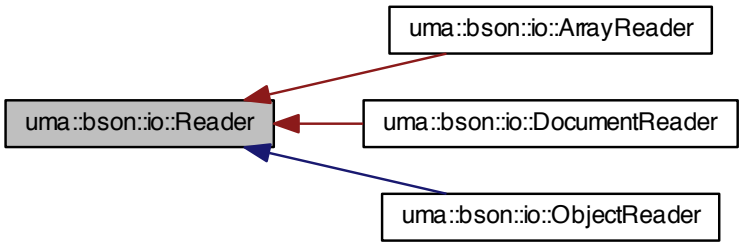
- /Users/rakesh/projects/opensource/mongoviewer/bson/src/api/uma/bson/PrimitiveValue.h

7.42 uma::bson::io::Reader Class Reference

A base streaming parser that parses a stream of BSON bytes from an input stream and transforms into a object model.

`#include <Reader.h>`

Inheritance diagram for uma::bson::io::Reader:



Public Member Functions

- virtual ~Reader ()

  Virtual destructor for sub-classes.

Protected Member Functions

- int32_t readInt (std::istream &fin)

  Reads the next 4 bytes of data from the input stream as a `int32_t` value.
- double readDouble (std::istream &fin)

  Reads the next 8 bytes of data from the input stream as a `double` value.
- int64_t readLong (std::istream &fin)

  Reads the next 8 bytes of data from the input stream as a `int64_t` value.
- void readString (std::istream &fin, std::string &str)

  Reads a `string` value from the stream as defined in the BSON specification.
- void readCharArray (std::istream &fin, std::string &str)

  Reads a `cstring` value from the stream as defined in the BSON specification.
- BinaryData readBinary (std::istream &fin)

  Reads a binary data object from the stream.
- ObjectId readOID (std::istream &fin)

  Reads the next 12 bytes from the stream into a MongoDB OID value.
- RegularExpression readRegEx (std::istream &fin)

  Reads a regular expression object from the stream.
- DatabaseReference readDbRef (std::istream &fin)

  Reads a MongoDB database reference instance from the stream.
- CodeWithScope readCodeWScope (std::istream &fin)

  Reads a code object with scope document from the stream.
- Timestamp readTimestamp (std::istream &fin)

  Reads the next 8 bytes from the stream into a MongoDB timestamp object.
- Document readDocument (std::istream &fin)

  Reads an embedded (or whole) document from the stream.
- Array readArray (std::istream &fin)

  Reads an embedded array from the stream.

7.42.1 Detailed Description

A base streaming parser that parses a stream of BSON bytes from an input stream and transforms into a object model.

Copyright

Copyright ©2012, Sans Pareil Technologies, Inc.

Date

Created 2012/09/24 11:42

Author

Rakesh

Version

Id:

Reader.h 181 2012-12-21 01:08:58Z spt

7.42.2 Constructor & Destructor Documentation

7.42.2.1 virtual uma::bson::io::Reader::~Reader ( ) [inline], [virtual]

Virtual destructor for sub-classes.

7.42.3 Member Function Documentation

7.42.3.1 Array uma::bson::io::Reader::readArray (std::istream & *fin*) [protected]

Reads an embedded array from the stream.

Parameters

fin	The input stream to read from.

Returns

Array The parsed array instance.

7.42.3.2 BinaryData uma::bson::io::Reader::readBinary (std::istream & *fin*) [protected]

Reads a binary data object from the stream.

Parameters

fin	The input stream from which the value is to be read.

Returns

BinaryData The binary data encoded in the stream.

7.42.3.3 void uma::bson::io::Reader::readCharArray (std::istream & *fin*, std::string & *str*) [protected]

Reads a `cstring` value from the stream as defined in the BSON specification.

As per the BSON specification `cstring` values are represented by stream of characters terminated by a `null-terminator` (similar to a char array in C).

Parameters

fin	The input stream to read from.
str	The string that is to be appended the data from the input stream

7.42.3.4 CodeWithScope uma::bson::io::Reader::readCodeWScope (std::istream & *fin*) [protected]

Reads a code object with scope document from the stream.

Parameters

fin	The input stream to read from

Returns

CodeWithScope The parsed code with scope object instance.

7.42.3.5 DatabaseReference uma::bson::io::Reader::readDbRef (std::istream & *fin*) [protected]

Reads a MongoDB database reference instance from the stream.

Parameters

fin	The input stream to read from

Returns

DatabaseReference The parsed database reference instance.

7.42.3.6 Document uma::bson::io::Reader::readDocument (std::istream & *fin*) [protected]

Reads an embedded (or whole) document from the stream.

The next 4 bytes of the stream indicate the total bytes that make up the document. Reads the data into a document instance.

Parameters

fin	The input stream to read from

Returns

Document The parsed document instance.

7.42.3.7 double uma::bson::io::Reader::readDouble (std::istream & *fin*) [protected]

Reads the next 8 bytes of data from the input stream as a `double` value.

Parameters

fin	The input stream from which the value is to be read.

Returns

double The value from the next 8 bytes in the stream.

7.42.3.8 int32_t uma::bson::io::Reader::readInt (std::istream & *fin*) [protected]

Reads the next 4 bytes of data from the input stream as a `int32_t` value.

Parameters

fin	The input stream from which the value is to be read.

Returns

int32_t The value from the next 4 bytes in the stream.

7.42.3.9 int64_t uma::bson::io::Reader::readLong (std::istream & *fin*) [protected]

Reads the next 8 bytes of data from the input stream as a `int64_t` value.

Parameters

fin	The input stream from which the value is to be read.

Returns

int64_t The value from the next 8 bytes in the stream.

7.42.3.10 ObjectId uma::bson::io::Reader::readOID (std::istream & *fin*) [protected]

Reads the next 12 bytes from the stream into a MongoDB OID value.

Parameters

fin	The input stream to read from.

Returns

OID The OID instance representing the next 12 bytes in the stream.

7.42.3.11 RegularExpression uma::bson::io::Reader::readRegEx (std::istream & *fin*) [protected]

Reads a regular expression object from the stream.

Parameters

fin	The input stream to read from

Returns

RegularExpression The parsed regular expression instance.

7.42.3.12 void uma::bson::io::Reader::readString (std::istream & *fin*, std::string & *str*) [protected]

Reads a `string` value from the stream as defined in the BSON specification.

As per the BSON specification `string` values are represented by a `int32_t` value that holds the total length of the string (including the null-terminator), followed by the stream of bytes. `string` instances may contain the null character within in.

Parameters

fin	The input stream to read from.
str	The string that is to be appended the data from the input stream

7.42.3.13 Timestamp uma::bson::io::Reader::readTimestamp (std::istream & *fin*) [protected]

Reads the next 8 bytes from the stream into a MongoDB timestamp object.

Parameters

fin	The input stream to read from.

Returns

Timestamp The parsed timestamp instance.

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

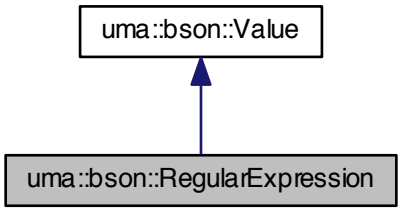
- /Users/rakesh/projects/opensource/mongoviewer/bson/src/api/uma/bson/io/Reader.h

7.43 uma::bson::RegularExpression Class Reference

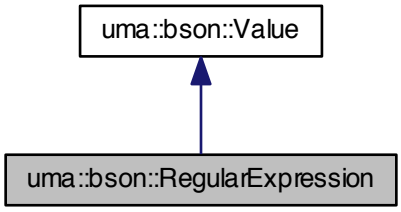
A POD that represents a regular expression type BSON element value.

`#include <RegularExpression.h>`

Inheritance diagram for uma::bson::RegularExpression:



Collaboration diagram for uma::bson::RegularExpression:



Public Member Functions

- RegularExpression ()

  Default CTOR.
- RegularExpression (const std::string &re, const std::string &f=std::string())

  Create a new instance that encapsulates the specified regular expression and flags values.
- const std::string & getRegex () const

  Return the regular expression value encapsulated in this instance as a constant reference.
- std::string & getRegex ()

  Return the regular expression value encapsulated in this instance as a non-constant reference.
- RegularExpression & setRegex (const std::string &value)

  Set the regular expression encapsulated in this instance.
- const std::string & getFlags () const

  Return the flags value encapsulated in this instance as a constant reference.
- std::string & getFlags ()

  Return the flags value encapsulated in this instance as a non-constant reference.
- RegularExpression & setFlags (const std::string &value)

  Set the regular expression flags value encapsulated in this instance.
- std::string toString () const

  Return a string representation of this instance.

- int32_t getSize () const

  Return the total size in bytes the BSON representation of the data in this instance occupies.
- Value::Type getType () const

  Returns the type for this instance as listed in the BSON specifications.

Additional Inherited Members

7.43.1 Detailed Description

A POD that represents a regular expression type BSON element value.

Date

Created 2012/09/09 21:41

Copyright

Author

Rakesh

Version

Id:

RegularExpression.h 172 2012-12-14 17:21:21Z spt

7.43.2 Constructor & Destructor Documentation

7.43.2.1 uma::bson::RegularExpression::RegularExpression ( ) [inline]

Default CTOR.

7.43.2.2 uma::bson::RegularExpression::RegularExpression (const std::string & *re*, const std::string & *f* = std::string()) [inline]

Create a new instance that encapsulates the specified regular expression and flags values.

Parameters

re	The regular expression to encapsulate
f	The flags to encapsulate. Defaults to empty.

7.43.3 Member Function Documentation

7.43.3.1 const std::string& uma::bson::RegularExpression::getFlags ( ) const [inline]

Return the flags value encapsulated in this instance as a constant reference.

Returns

The regex flags.

7.43.3.2 std::string& uma::bson::RegularExpression::getFlags ( ) [inline]

Return the flags value encapsulated in this instance as a non-constant reference.

Returns

The regex flags.

7.43.3.3 const std::string& uma::bson::RegularExpression::getRegex ( ) const [inline]

Return the regular expression value encapsulated in this instance as a constant reference.

Returns

The regular expression value.

7.43.3.4 std::string& uma::bson::RegularExpression::getRegex ( ) [inline]

Return the regular expression value encapsulated in this instance as a non-constant reference.

Returns

The regular expression value.

7.43.3.5 int32_t uma::bson::RegularExpression::getSize ( ) const [inline], [virtual]

Return the total size in bytes the BSON representation of the data in this instance occupies.

Returns

Returns length in bytes

Implements uma::bson::Value.

7.43.3.6 Value::Type uma::bson::RegularExpression::getType ( ) const [inline], [virtual]

Returns the type for this instance as listed in the BSON specifications.

Returns

Returns uma::bson::Value::Type::RegEx

Implements uma::bson::Value.

7.43.3.7 RegularExpression& uma::bson::RegularExpression::setFlags (const std::string & *value*) [inline]

Set the regular expression flags value encapsulated in this instance.

Parameters

value	The regex flags value

Returns

This instance for method chaining.

7.43.3.8 RegularExpression& uma::bson::RegularExpression::setRegex (const std::string & *value*) [inline]

Set the regular expression encapsulated in this instance.

Parameters

value	The regular expression value

Returns

This instance for method chaining

7.43.3.9 std::string uma::bson::RegularExpression::toString ( ) const [inline]

Return a string representation of this instance.

The returned value is primarily intended for logging/debugging purposes.

Returns

The string representation

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

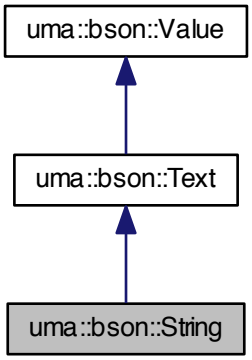
- /Users/rakesh/projects/opensource/mongoviewer/bson/src/api/uma/bson/RegularExpression.h

7.44 uma::bson::String Class Reference

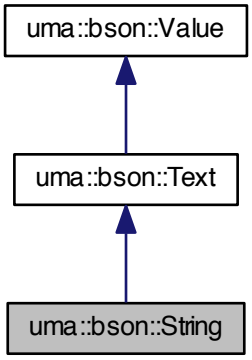
A POD that represents a string value in a BSON element.

`#include <String.h>`

Inheritance diagram for uma::bson::String:



Collaboration diagram for uma::bson::String:



Public Member Functions

- String ()

  Default CTOR.
- String (const std::string &v)

  Create a new instance that stores the specified value.
- String (const char *data)

  Create a new instance from a C-style char array.
- Value::Type getType () const

  Returns the type for this instance as listed in the BSON specifications.

Additional Inherited Members

7.44.1 Detailed Description

A POD that represents a string value in a BSON element.

Date

Created 2012/09/05 19:55

Copyright

Author

Rakesh

Version

Id:

String.h 172 2012-12-14 17:21:21Z spt

7.44.2 Constructor & Destructor Documentation

7.44.2.1 uma::bson::String::String ( ) [inline]

Default CTOR.

7.44.2.2 uma::bson::String::String (const std::string & *v*) [inline]

Create a new instance that stores the specified value.

Parameters

v	The string value to store

7.44.2.3 uma::bson::String::String (const char * *data*) [inline]

Create a new instance from a C-style char array.

Parameters

data	The string value to store

7.44.3 Member Function Documentation

7.44.3.1 Value::Type uma::bson::String::getType ( ) const [inline], [virtual]

Returns the type for this instance as listed in the BSON specifications.

Returns

Returns uma::bson::Value::Type::String

Implements uma::bson::Value.

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

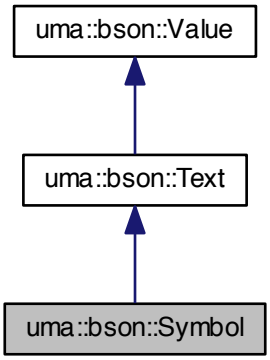
- /Users/rakesh/projects/opensource/mongoviewer/bson/src/api/uma/bson/String.h

7.45 uma::bson::Symbol Class Reference

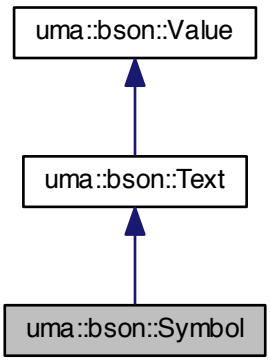
A POD that represents a programming language symbol type BSON element value.

`#include <Symbol.h>`

Inheritance diagram for uma::bson::Symbol:



Collaboration diagram for uma::bson::Symbol:



Public Member Functions

- Symbol ()

  Default CTOR.
- Symbol (const std::string &v)

  Create a new instance that encapsulates the specified value.
- Symbol (const char *data)

  Create a new instance from a C-style char array.
- Value::Type getType () const

  Returns the type for this instance as listed in the BSON specifications.

Additional Inherited Members

7.45.1 Detailed Description

A POD that represents a programming language symbol type BSON element value.

Date

Created 2012/09/09 21:49

Copyright

Author

Rakesh

Version

Id:

Symbol.h 172 2012-12-14 17:21:21Z spt

7.45.2 Constructor & Destructor Documentation

7.45.2.1 uma::bson::Symbol::Symbol ( ) [inline]

Default CTOR.

7.45.2.2 uma::bson::Symbol::Symbol (const std::string & *v*) [inline]

Create a new instance that encapsulates the specified value.

Parameters

v	The string value to encapsulate

7.45.2.3 uma::bson::Symbol::Symbol (const char * *data*) [inline]

Create a new instance from a C-style char array.

Parameters

data	The string value to encapsulate

7.45.3 Member Function Documentation

7.45.3.1 Value::Type uma::bson::Symbol::getType ( ) const [inline], [virtual]

Returns the type for this instance as listed in the BSON specifications.

Returns

Returns uma::bson::Value::Type::Symbol

Implements uma::bson::Value.

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

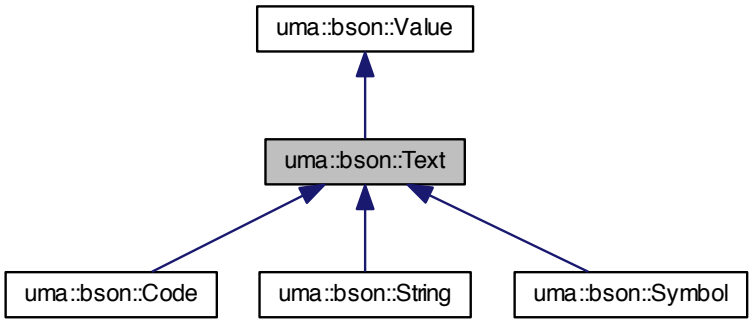
- /Users/rakesh/projects/opensource/mongoviewer/bson/src/api/uma/bson/Symbol.h

7.46 uma::bson::Text Class Reference

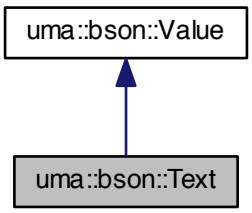
A class that holds a string. BSON string/text values can be large, hence take care when copy constructing or assigning text instances.

`#include <Text.h>`

Inheritance diagram for uma::bson::Text:



Collaboration diagram for uma::bson::Text:



Public Member Functions

- const std::string & getValue () const

  Return the value held in this instance as a constant reference.
- std::string & getValue ()

  Return the value held in this instance as a reference.
- Text & setValue (const std::string &text)

  Set the value held in this instance.
- Text & operator+= (const std::string &str)

  Append the specified string to the value held in this instance.
- operator std::string () const

  Cast operator to make this instance behave as a standard string.
- int32_t getSize () const

  Return the size in bytes for the BSON representation of this instance.
- size_t size () const

  Return the size of the held string. As with std::string the returned size does not include a null-terminator character.

Protected Member Functions

- Text ()

  Default CTOR. Use to create an empty string value.
- Text (const std::string &text)

  Create a new text instance that represents the specified string value.
- Text (const char *data)

  Create a text instance from a C-style char array.

Additional Inherited Members

7.46.1 Detailed Description

A class that holds a string. BSON string/text values can be large, hence take care when copy constructing or assigning text instances.

Date

Created 2012/11/30 6:10

Copyright

Copyright ©2012, Sans Pareil Technologies, Inc.

Author

Rakesh

Version

Id:

Text.h 174 2012-12-15 13:13:56Z spt

7.46.2 Constructor & Destructor Documentation

7.46.2.1 uma::bson::Text::Text ( ) [inline], [protected]

Default CTOR. Use to create an empty string value.

This is the recommended way to create and populate string values. Fetch the reference to the held string value and update/populate it.

7.46.2.2 uma::bson::Text::Text (const std::string & *text*) [inline], [explicit], [protected]

Create a new text instance that represents the specified string value.

Warning

This form will involve a copy of the passed in string value. The recommended way to populate a text instance is by default constructing an instance, and then populating the held string value.

Parameters

text	The string value to hold in this instance.

7.46.2.3 uma::bson::Text::Text (const char * *data*) [inline], [explicit], [protected]

Create a text instance from a C-style char array.

Parameters

data	The string value to hold

7.46.3 Member Function Documentation

7.46.3.1 int32_t uma::bson::Text::getSize ( ) const [inline], [virtual]

Return the size in bytes for the BSON representation of this instance.

Returns

Returns `4` for the length of the string + size of the string + `1` for the null terminator

Implements uma::bson::Value.

7.46.3.2 const std::string& uma::bson::Text::getValue ( ) const [inline]

Return the value held in this instance as a constant reference.

Returns

The constant reference to the string.

7.46.3.3 std::string& uma::bson::Text::getValue ( ) [inline]

Return the value held in this instance as a reference.

Returns

The reference to the string.

7.46.3.4 uma::bson::Text::operator std::string ( ) const [inline]

Cast operator to make this instance behave as a standard string.

7.46.3.5 Text& uma::bson::Text::operator+= (const std::string & *str*) [inline]

Append the specified string to the value held in this instance.

Parameters

str	The string to append.

Returns

This instance for method chaining

7.46.3.6 Text& uma::bson::Text::setValue (const std::string & *text*) [inline]

Set the value held in this instance.

Parameters

text	The new value to hold

Returns

This instance for method chaining

7.46.3.7 size_t uma::bson::Text::size ( ) const [inline]

Return the size of the held string. As with std::string the returned size does not include a null-terminator character.

Returns

The size of the held string.

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

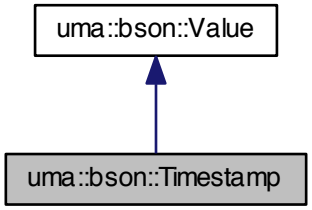
- /Users/rakesh/projects/opensource/mongoviewer/bson/src/api/uma/bson/Text.h

7.47 uma::bson::Timestamp Class Reference

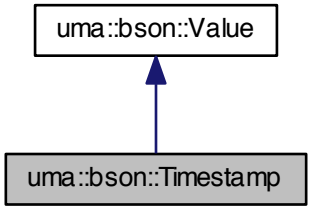
A POD that represents a database timestamp BSON element value.

`#include <Timestamp.h>`

Inheritance diagram for uma::bson::Timestamp:



Collaboration diagram for uma::bson::Timestamp:



Public Member Functions

- Timestamp ()

  Default CTOR.
- Timestamp (const std::time_t seconds, const int32_t inc)

  Create a new instance with the specified increment and time values.
- const Poco::Timestamp & getTime () const

  Return the time value in this instance as a constant reference.

- Poco::Timestamp & getTime ()

  Return the time value in this instance as a non-constant reference.
- Timestamp & setTime (const Poco::Timestamp &ts)

  Set the time value encapsulated in this instance.
- Timestamp & setTime (const std::time_t seconds)

  Set the time value encapsulated in this instance.
- int32_t getIncrement () const

  Returns the increment value encapsulated in this instance.
- Timestamp & setIncrement (const int32_t inc)

  Set the increment value encapsulated in this instance.
- std::string toString () const

  Return a string representation of the data encapsulated in this instance primarily intended for logging/debugging purposes.
- int32_t getSize () const

  Return the size in bytes of the BSON representation of this instance.
- Value::Type getType () const

  Returns the type for this instance as listed in the BSON specifications.

Additional Inherited Members

7.47.1 Detailed Description

A POD that represents a database timestamp BSON element value.

Timestamp values are represented by an 8 byte long number. The first four bytes are used to represent the increment value, and the next four bytes represent the seconds since UNIX epoch time.

Date

Created 2012/09/09 21:55

Copyright

Author

Rakesh

Version

Id:

Timestamp.h 172 2012-12-14 17:21:21Z spt

7.47.2 Constructor & Destructor Documentation

7.47.2.1 uma::bson::Timestamp::Timestamp ( ) [inline]

Default CTOR.

7.47.2.2 uma::bson::Timestamp::Timestamp (const std::time_t *seconds*, const int32_t *inc*) [inline]

Create a new instance with the specified increment and time values.

Parameters

seconds	The time since UNIX epoch to encapsulate
inc	The increment value to encapsulate

7.47.3 Member Function Documentation

7.47.3.1 int32_t uma::bson::Timestamp::getIncrement ( ) const [inline]

Returns the increment value encapsulated in this instance.

Returns

The increment value

7.47.3.2 int32_t uma::bson::Timestamp::getSize ( ) const [inline], [virtual]

Return the size in bytes of the BSON representation of this instance.

Returns

Returns 8 as per BSON specifications

Implements uma::bson::Value.

7.47.3.3 const Poco::Timestamp& uma::bson::Timestamp::getTime ( ) const [inline]

Return the time value in this instance as a constant reference.

Returns

The time value.

7.47.3.4 Poco::Timestamp& uma::bson::Timestamp::getTime ( ) [inline]

Return the time value in this instance as a non-constant reference.

Returns

The time value.

7.47.3.5 Value::Type uma::bson::Timestamp::getType ( ) const [inline], [virtual]

Returns the type for this instance as listed in the BSON specifications.

Returns

Returns uma::bson::Value::Type::Timestamp

Implements uma::bson::Value.

7.47.3.6 Timestamp& uma::bson::Timestamp::setIncrement (const int32_t *inc*) [inline]

Set the increment value encapsulated in this instance.

Parameters

inc	The new increment value

Returns

This instance for method chaining

7.47.3.7 Timestamp& uma::bson::Timestamp::setTime (const Poco::Timestamp & *ts*) [inline]

Set the time value encapsulated in this instance.

Parameters

ts	The new value to set

Returns

This instance for method chaining

7.47.3.8 Timestamp& uma::bson::Timestamp::setTime (const std::time_t *seconds*) [inline]

Set the time value encapsulated in this instance.

Parameters

seconds	The seconds since UNIX epoch value to set.

Returns

This instance for method chaining

7.47.3.9 std::string uma::bson::Timestamp::toString ( ) const [inline]

Return a string representation of the data encapsulated in this instance primarily intended for logging/debugging purposes.

Returns

Textual representation of the data.

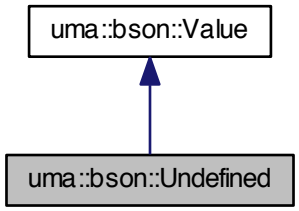
The documentation for this class was generated from the following file:

- /Users/rakesh/projects/opensource/mongoviewer/bson/src/api/uma/bson/Timestamp.h

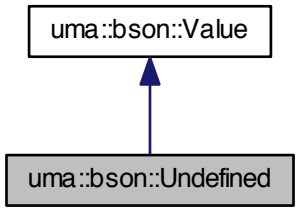
7.48 uma::bson::Undefined Class Reference

`#include <Undefined.h>`

Inheritance diagram for uma::bson::Undefined:



Collaboration diagram for uma::bson::Undefined:



Public Member Functions

- Undefined ()

  Default CTOR.
- int32_t getSize () const

  Return the size in bytes for the BSON representation of this value.
- Value::Type getType () const

  Returns the type for this instance as listed in the BSON specifications.

Additional Inherited Members

7.48.1 Detailed Description

A POD that represents a BSON element of type undefined.

Date

Created 2012/09/06 18:55

Copyright

Copyright ©2012, Sans Pareil Technologies, Inc.

Author

Rakesh

Version

Id:

Undefined.h 172 2012-12-14 17:21:21Z spt

7.48.2 Constructor & Destructor Documentation

7.48.2.1 uma::bson::Undefined::Undefined ( ) [inline]

Default CTOR.

7.48.3 Member Function Documentation

7.48.3.1 int32_t uma::bson::Undefined::getSize ( ) const [inline], [virtual]

Return the size in bytes for the BSON representation of this value.

Returns

Returns `0` as per specifications.

Implements uma::bson::Value.

7.48.3.2 Value::Type uma::bson::Undefined::getType ( ) const [inline], [virtual]

Returns the type for this instance as listed in the BSON specifications.

Returns

Returns uma::bson::Value::Type::Undefined

Implements uma::bson::Value.

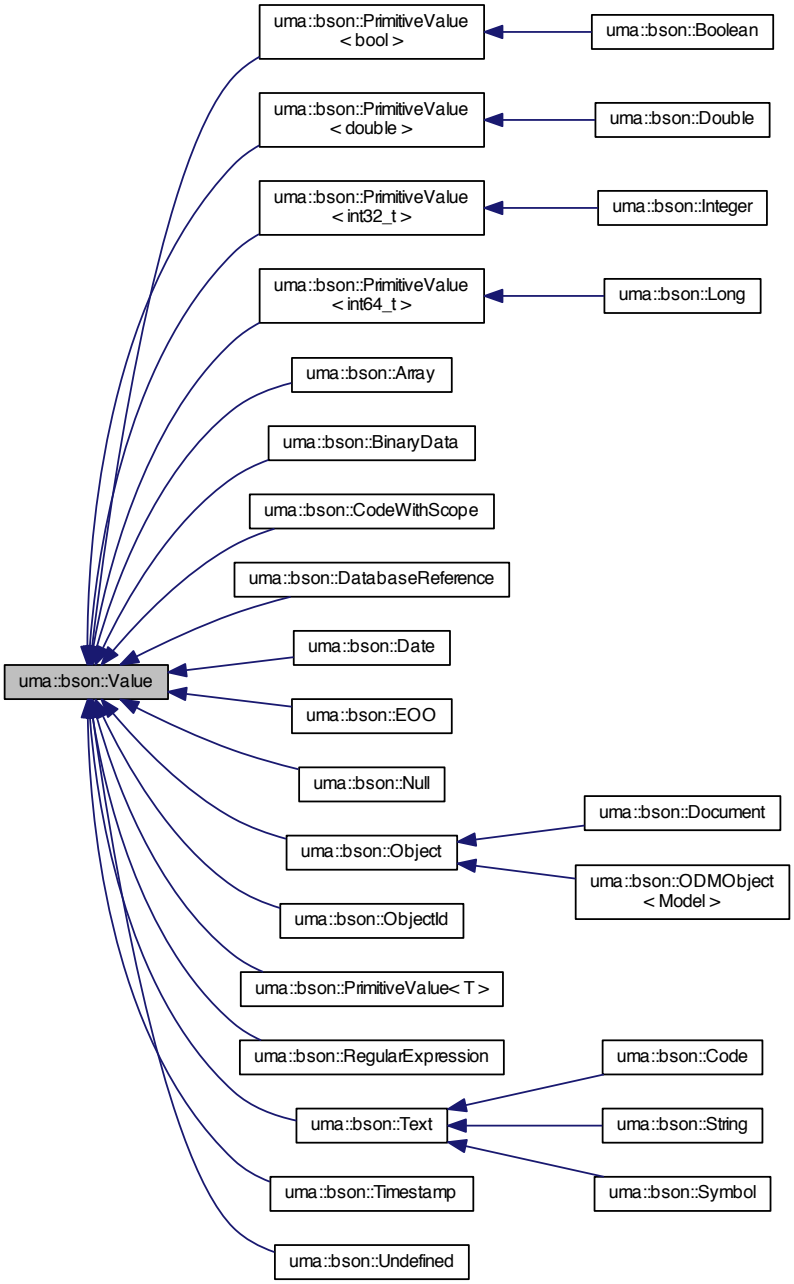
The documentation for this class was generated from the following file:

- /Users/rakesh/projects/opensource/mongoviewer/bson/src/api/uma/bson/Undefined.h

7.49 uma::bson::Value Class Reference

`#include <Value.h>`

Inheritance diagram for uma::bson::Value:



Public Types

- enum Type {
  Eoo = 0, Double = 1, String = 2, Object = 3,
  Array = 4, BinData = 5, Undefined = 6, OID = 7,
  Boolean = 8, Date = 9, Null = 10, RegEx = 11,
  DbRef = 12, Code = 13, Symbol = 14, CodeWScope = 15,
  Integer = 16, Timestamp = 17, Long = 18 }

Public Member Functions

- virtual ~Value ()

  Virtual DTOR for sub-classes.
- virtual Type getType () const =0

  Return the type for this instance as listed in the BSON specifications.
- virtual int32_t getSize () const =0

  Return the size of the data as per BSON specifications.
- const std::string & getTypeName () const

  Returns a standard textual value for this type.

Static Public Member Functions

- static const std::string & getTypeName (const Type type)

  Return a descriptive text about the specified type.

Protected Member Functions

- Value ()

7.49.1 Detailed Description

The abstract base class that represents a BSON data type. Concrete implementations are used to represent the 18 types of data specified in the BSON specifications.

Date

Created 2012/09/05 19:55

Copyright

Author

Rakesh

Version

Id:

Value.h 185 2012-12-21 15:39:00Z spt

7.49.2 Member Enumeration Documentation

7.49.2.1 enum uma::bson::Value::Type

BSON element/field types.

Enumerator

Eoo End of document/element.

Double Floating point.
String UTF-8 string.
Object Embedded document.
Array Embedded array.
BinData Binary data.
Undefined Undefined — Deprecated.
OID MongoDB ObjectId.
Boolean Boolean false: 0, true: 1.
Date UTC datetime - milliseconds since UNIX epoch time.
Null Null value.
RegEx Regular expression.
DbRef DBPointer — Deprecated.
Code JavaScript code.
Symbol Symbol — Deprecated.
CodeWScope JavaScript code w/ scope.
Integer 32-bit Integer
Timestamp Timestamp - increment: 4bytes, time: 4bytes.
Long 64-bit integer

7.49.3 Constructor & Destructor Documentation

7.49.3.1 virtual uma::bson::Value::~Value ( ) [inline], [virtual]

Virtual DTOR for sub-classes.

7.49.3.2 uma::bson::Value::Value ( ) [inline], [protected]

7.49.4 Member Function Documentation

7.49.4.1 virtual int32_t uma::bson::Value::getSize ( ) const [pure virtual]

Return the size of the data as per BSON specifications.

Returns

The size of the BSON data.

Implemented in uma::bson::Document, uma::bson::Array, uma::bson::Object, uma::bson::BinaryData, uma::bson::Timestamp, uma::bson::RegularExpression, uma::bson::DatabaseReference, uma::bson::ObjectId, uma::bson::CodeWithScope, uma::bson::Date, uma::bson::Text, uma::bson::Boolean, uma::bson::Double, uma::bson::Integer, uma::bson::Long, uma::bson::EOO, uma::bson::Undefined, and uma::bson::Null.

7.49.4.2 virtual Type uma::bson::Value::getType ( ) const [pure virtual]

Return the type for this instance as listed in the BSON specifications.

Returns

Implementations will return the appropriate enum value for its BSON data type.

Implemented in uma::bson::Array, uma::bson::Object, uma::bson::BinaryData, uma::bson::Timestamp, uma::bson::RegularExpression, uma::bson::DatabaseReference, uma::bson::ObjectId, uma::bson::CodeWithScope, uma::bson::Date, uma::bson::Boolean, uma::bson::Double, uma::bson::Integer, uma::bson::Long, uma::bson::Code, uma::bson::Symbol, uma::bson::String, uma::bson::EOO, uma::bson::Undefined, and uma::bson::Null.

7.49.4.3 const std::string& uma::bson::Value::getTypeName ( ) const [inline]

Returns a standard textual value for this type.

Returns

A text value representing for the document type. Useful for logging/debugging

7.49.4.4 static const std::string& uma::bson::Value::getTypeName (const Type *type*) [static]

Return a descriptive text about the specified type.

Parameters

type	The type to describe

Returns

The textual description of the type.

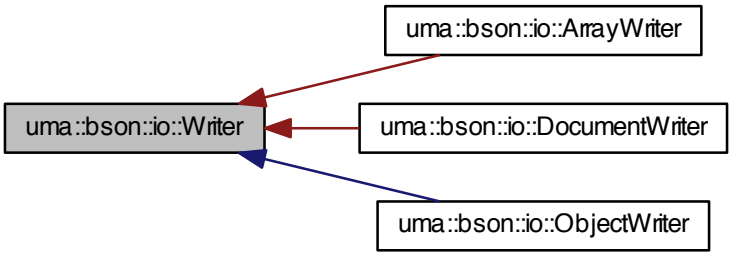
The documentation for this class was generated from the following file:

- /Users/rakesh/projects/opensource/mongoviewer/bson/src/api/uma/bson/Value.h

7.50 uma::bson::io::Writer Class Reference

`#include <Writer.h>`

Inheritance diagram for uma::bson::io::Writer:



Public Member Functions

- virtual ~Writer ()

  Virtual destructor for sub-classes.

Protected Member Functions

- Writer (std::ostream &os)

Create a new instance of the writer that will write to the specified output stream.

- std::ostream & getStream ()

  Return the output stream for use from sub-classes.
- void write (const int32_t value)

  Writes a 32 bit integer value to the output stream.
- void write (const int64_t value)

  Writes a 64 bit integer value to the output stream.
- void write (const double value)

  Writes a `double` value to the output stream.
- void write (const bool value)

  Writes a `boolean` value to the output stream.
- void write (const std::string &value, const bool withLength=false)

  Writes a `std::string` value to the output stream.
- void write (const BinaryData &data)

  Writes binary data type value to the output stream.
- void write (const ObjectId &oid)

  Writes a MongoDB OID value to the output stream.
- void write (const Date &date)

  Writes a date-time value to the output stream.
- void write (const RegularExpression &regex)

  Writes a regular expression value to the output stream.
- void write (const DatabaseReference &dbref)

  Writes a MongoDB database reference value to the output stream.
- void write (const CodeWithScope &code)

  Writes a code with scope value to the output stream.
- void write (const Timestamp &ts)

  Writes a timestamp value to the output stream.
- void write (const Element &element)

  Serialises a bson element to the output stream.
- void write (const Document &doc)

  Serialises a document to the output stream.
- virtual void write (const Array &arr)

  Serialises an array to the output stream.

7.50.1 Detailed Description

A base streaming writer class that transforms the data in a object model into a BSON representation and writes it to an output stream.

Date

Created 2012/09/21 19:49

Copyright

Author

Rakesh

Version

Id:

Writer.h 181 2012-12-21 01:08:58Z spt

7.50.2 Constructor & Destructor Documentation

7.50.2.1 virtual uma::bson::io::Writer::~Writer ( ) [inline], [virtual]

Virtual destructor for sub-classes.

7.50.2.2 uma::bson::io::Writer::Writer (std::ostream & *os*) [inline], [protected]

Create a new instance of the writer that will write to the specified output stream.

Parameters

os	The output stream to write the BSON representation to

7.50.3 Member Function Documentation

7.50.3.1 std::ostream& uma::bson::io::Writer::getStream ( ) [inline], [protected]

Return the output stream for use from sub-classes.

Returns

std::ostream The output stream to which the BSON output is being written.

7.50.3.2 void uma::bson::io::Writer::write (const int32_t *value*) [protected]

Writes a 32 bit integer value to the output stream.

Parameters

value	The value to serialise as BSON

7.50.3.3 void uma::bson::io::Writer::write (const int64_t *value*) [protected]

Writes a 64 bit integer value to the output stream.

Parameters

value	The value to serialise to BSON

7.50.3.4 void uma::bson::io::Writer::write (const double *value*) [protected]

Writes a `double` value to the output stream.

Parameters

value	The value to serialise to BSON

7.50.3.5 void uma::bson::io::Writer::write (const bool *value*) [protected]

Writes a `boolean` value to the output stream.

Parameters

value	The value to serialise to BSON

7.50.3.6 void uma::bson::io::Writer::write (const std::string & *value,* const bool *withLength* = `false`) [protected]

Writes a `std::string` value to the output stream.

Used to serialise a std::string as BSON to the output stream. This method is used to output both `string` and `cstring` values as referred to in the BSON specifications.

Parameters

value	The value to serialise to BSON
withLength	Flag that indicates whether a `string` (`true`) or a `cstring` (`false`) is to be written.

7.50.3.7 void uma::bson::io::Writer::write (const **BinaryData** & *data*) [protected]

Writes binary data type value to the output stream.

Parameters

data	The value to serialise to BSON

7.50.3.8 void uma::bson::io::Writer::write (const **ObjectId** & *oid*) [protected]

Writes a MongoDB OID value to the output stream.

Parameters

oid	The value to serialise to BSON

7.50.3.9 void uma::bson::io::Writer::write (const **Date** & *date*) [protected]

Writes a date-time value to the output stream.

Date-time values are serialised as an `int64_t` value that represents the milliseconds since UNIX epoch time - January 1, 1970.

Parameters

date	The value to serialise to BSON

7.50.3.10 void uma::bson::io::Writer::write (const **RegularExpression** & *regex*) [protected]

Writes a regular expression value to the output stream.

Parameters

regex	The value to serialise to BSON

7.50.3.11 void uma::bson::io::Writer::write (const DatabaseReference & *dbref*) [protected]

Writes a MongoDB database reference value to the output stream.

Parameters

dbref	The value to serialise to BSON

7.50.3.12 void uma::bson::io::Writer::write (const CodeWithScope & *code*) [protected]

Writes a code with scope value to the output stream.

Parameters

code	The value to serialise to BSON

7.50.3.13 void uma::bson::io::Writer::write (const Timestamp & *ts*) [protected]

Writes a timestamp value to the output stream.

Timestamps are written as a `int64_t` value, with the first 4 bytes representing an increment value, and the last 4 bytes representing a std::time_t value.

Parameters

ts	The value to serialise to BSON

7.50.3.14 void uma::bson::io::Writer::write (const Element & *element*) [protected]

Serialises a bson element to the output stream.

Determines the uma::bson::Element::getType() value for the element and delegates to the appropriate `write` method for serialising the value.

Parameters

element	The element to serialise to BSON

7.50.3.15 void uma::bson::io::Writer::write (const Document & *doc*) [protected]

Serialises a document to the output stream.

This method is also used to serialise embedded document values. Embedded document values are also included in values of type uma::bson::CodeWithScope and uma::bson::BinaryData.

Parameters

value	The value to serialise to BSON

7.50.3.16 virtual void uma::bson::io::Writer::write (const Array & *arr*) [protected], [virtual]

Serialises an array to the output stream.

Parameters

value	The value to serialise to BSON

Reimplemented in uma::bson::io::ObjectWriter.

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

- /Users/rakesh/projects/opensource/mongoviewer/bson/src/api/uma/bson/io/Writer.h

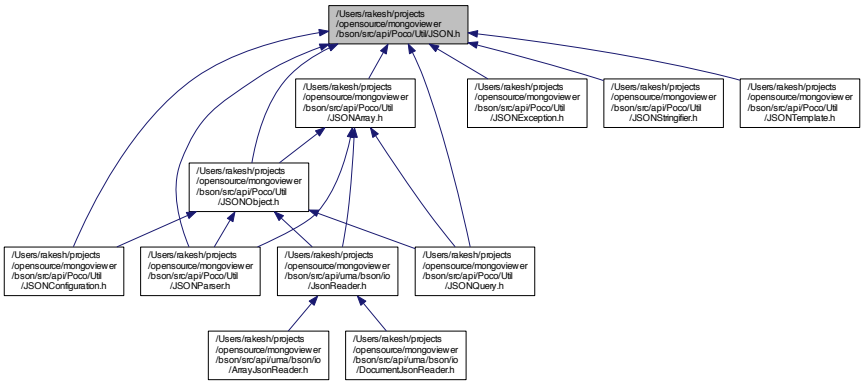
Chapter 8

File Documentation

8.1 /Users/rakesh/projects/opensource/mongoviewer/bson/src/api/mainpage.dox File Reference

8.2 /Users/rakesh/projects/opensource/mongoviewer/bson/src/api/Poco/Util/JSON.h File Reference

This graph shows which files directly or indirectly include this file:



Macros

- #define JSON_API

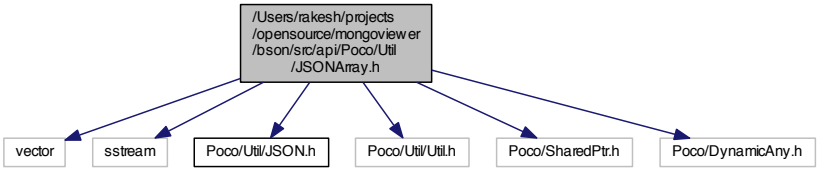
8.2.1 Macro Definition Documentation

8.2.1.1 #define JSON_API

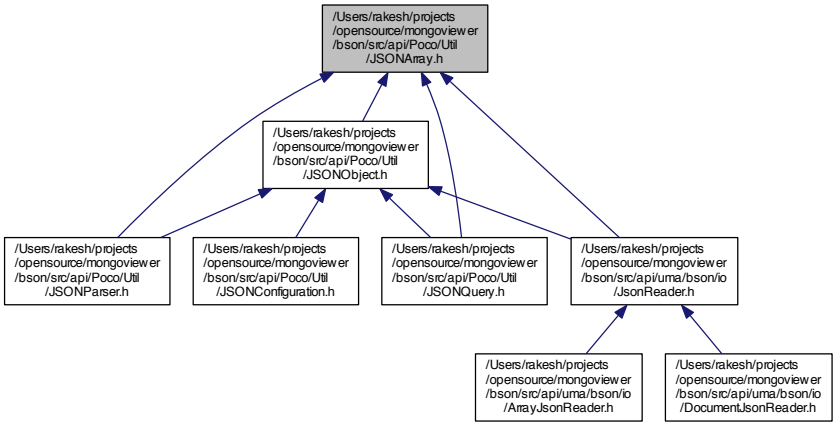
8.3 /Users/rakesh/projects/opensource/mongoviewer/bson/src/api/Poco/Util/JSONArray.h File Reference

```
#include <vector>
#include <sstream>
#include <Poco/Util/JSON.h>
#include <Poco/Util/Util.h>
#include <Poco/SharedPtr.h>
#include <Poco/DynamicAny.h>
```

Include dependency graph for JSONArray.h:



This graph shows which files directly or indirectly include this file:



Classes

- class Poco::Util::JSONArray
- class Poco::DynamicAnyHolderImpl< Util::JSONArray::Ptr >

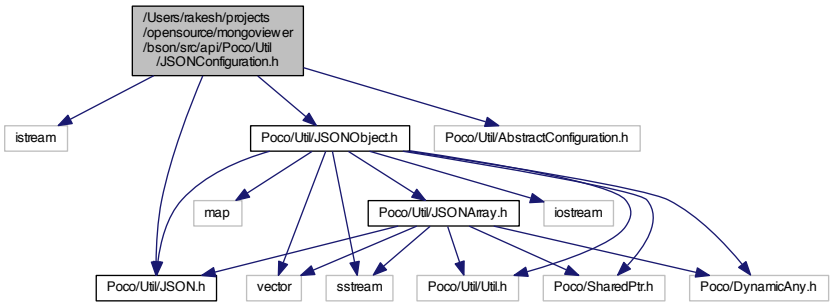
Namespaces

- namespace Poco
- namespace Poco::Util

8.4 /Users/rakesh/projects/opensource/mongoviewer/bson/src/api/Poco/Util/JSONConfiguration.h File Reference

```
#include <istream>
#include <Poco/Util/JSON.h>
#include <Poco/Util/AbstractConfiguration.h>
#include <Poco/Util/JSONObject.h>
```

Include dependency graph for JSONConfiguration.h:



Classes

- class Poco::Util::JSONConfiguration

  This configuration class extracts configuration properties from a JSON object. An XPath-like syntax for property names is supported to allow full access to the JSON object.

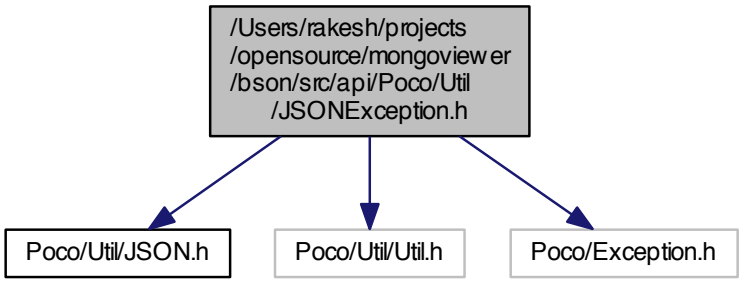
Namespaces

- namespace Poco
- namespace Poco::Util

8.5 /Users/rakesh/projects/opensource/mongoviewer/bson/src/api/Poco/Util/JSONException.h File Reference

```
#include <Poco/Util/JSON.h>
#include <Poco/Util/Util.h>
#include <Poco/Exception.h>
```

Include dependency graph for JSONException.h:



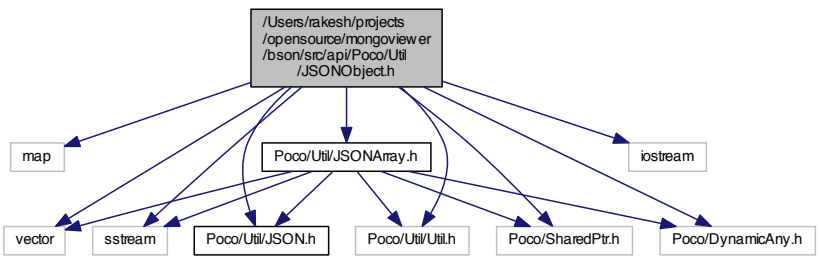
Namespaces

- namespace Poco
- namespace Poco::Util

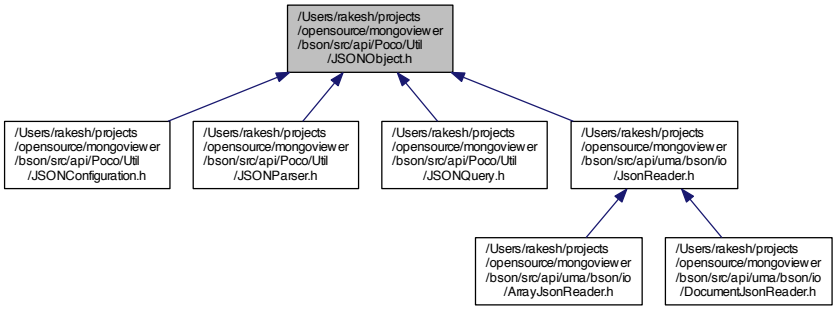
8.6 /Users/rakesh/projects/opensource/mongoviewer/bson/src/api/Poco/Util/JSONObject.h File Reference

```
#include <map>
#include <vector>
#include <iostream>
#include <sstream>
#include <Poco/SharedPtr.h>
#include <Poco/DynamicAny.h>
#include <Poco/Util/JSON.h>
#include <Poco/Util/Util.h>
#include <Poco/Util/JSONArray.h>
```

Include dependency graph for JSONObject.h:



This graph shows which files directly or indirectly include this file:



Classes

- class Poco::Util::JSONObject

  Represents a JSON object.
- class Poco::DynamicAnyHolderImpl< Util::JSONObject::Ptr >

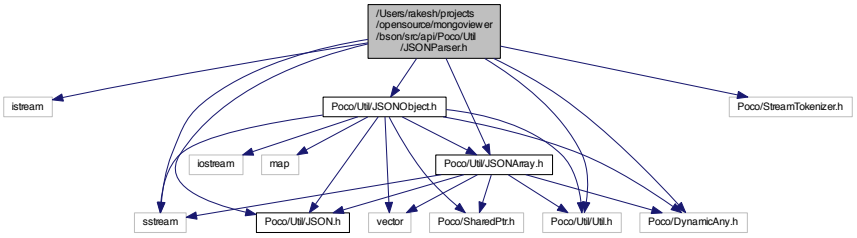
Namespaces

- namespace Poco
- namespace Poco::Util

8.7 /Users/rakesh/projects/opensource/mongoviewer/bson/src/api/Poco/Util/JSONParser.h File Reference

```
#include <istream>
#include <sstream>
#include <Poco/DynamicAny.h>
#include <Poco/StreamTokenizer.h>
#include <Poco/Util/JSON.h>
#include <Poco/Util/Util.h>
#include <Poco/Util/JSONObject.h>
#include <Poco/Util/JSONArray.h>
```

Include dependency graph for JSONParser.h:



Classes

- class Poco::Util::JSONParser

  A class for passing JSON strings or streams.

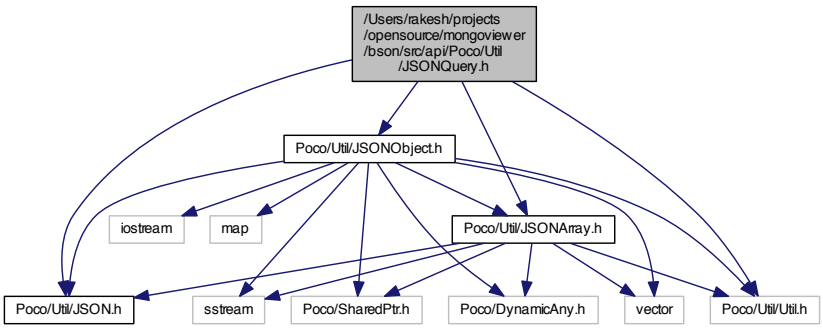
Namespaces

- namespace Poco
- namespace Poco::Util

8.8 /Users/rakesh/projects/opensource/mongoviewer/bson/src/api/Poco/Util/JSONQuery.h File Reference

```
#include <Poco/Util/JSON.h>
#include <Poco/Util/Util.h>
#include <Poco/Util/JSONObject.h>
#include <Poco/Util/JSONArray.h>
```

Include dependency graph for JSONQuery.h:



Classes

- class Poco::Util::JSONQuery

  Class that can be used to search for a value in a JSON object or array.

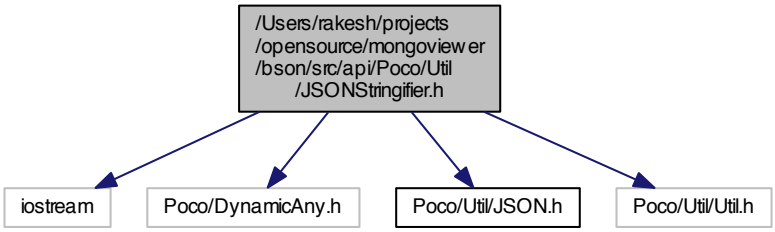
Namespaces

- namespace Poco
- namespace Poco::Util

8.9 /Users/rakesh/projects/opensource/mongoviewer/bson/src/api/Poco/Util/JSONStringifier.h File Reference

```
#include <iostream>
#include <Poco/DynamicAny.h>
#include <Poco/Util/JSON.h>
#include <Poco/Util/Util.h>
```

Include dependency graph for JSONStringifier.h:



Classes

- class Poco::Util::JSONStringifier

  Helper class for creating a String from a JSON object or array.

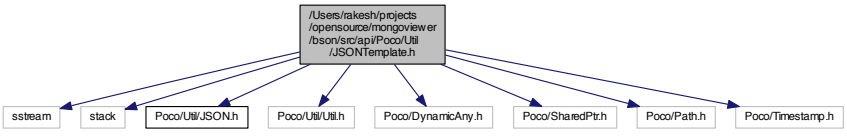
Namespaces

- namespace Poco
- namespace Poco::Util

8.10 /Users/rakesh/projects/opensource/mongoviewer/bson/src/api/Poco/Util/JSONTemplate.h File Reference

```
#include <sstream>
#include <stack>
#include <Poco/Util/JSON.h>
#include <Poco/Util/Util.h>
#include <Poco/DynamicAny.h>
#include <Poco/SharedPtr.h>
#include <Poco/Path.h>
#include <Poco/Timestamp.h>
```

Include dependency graph for JSONTemplate.h:



Classes

- class Poco::Util::JSONTemplate

  JSONTemplate is a template engine which uses JSON as input for generating output. There are commands for looping over JSON arrays, include other templates, conditional output, ...

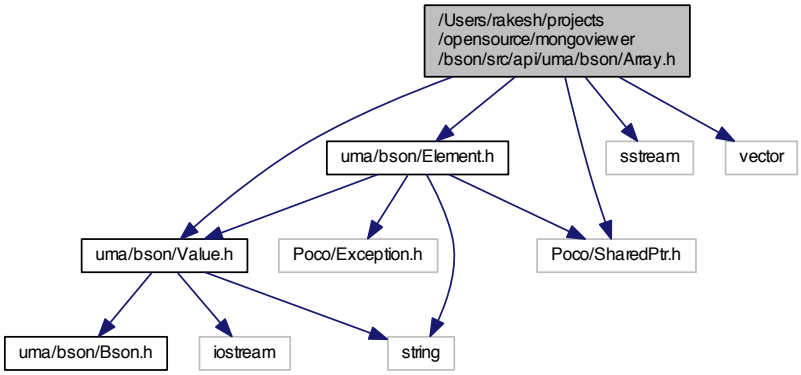
Namespaces

- namespace Poco
- namespace Poco::Util

8.11 /Users/rakesh/projects/opensource/mongoviewer/bson/src/api/uma/bson/Array.h File Reference

```
#include <uma/bson/Value.h>
#include <uma/bson/Element.h>
#include <Poco/SharedPtr.h>
#include <sstream>
#include <vector>
```

Include dependency graph for Array.h:



Classes

- class uma::bson::Array

  A class that represents a BSON array type document.

Namespaces

- namespace uma
- namespace uma::bson

  Classes that present a DOM style view of a BSON document.

Functions

- UMA_BSON_API bool uma::bson::operator== (const Array &lhs, const Array &rhs)

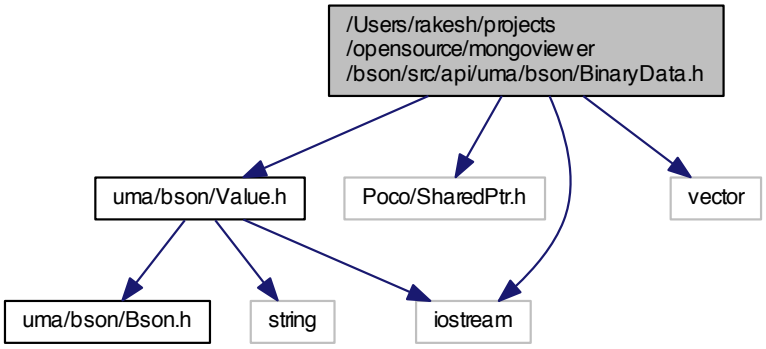
  Compare two arrays for equality. Arrays are considered equal if they have the same size and the same elements at the same indices in the array.
- bool uma::bson::operator!= (const Array &lhs, const Array &rhs)

  Just the reverse of operator ==.

8.12 /Users/rakesh/projects/opensource/mongoviewer/bson/src/api/uma/bson/BinaryData.h File Reference

```
#include <uma/bson/Value.h>
#include <Poco/SharedPtr.h>
#include <iostream>
#include <vector>
```

Include dependency graph for BinaryData.h:



Classes

- class uma::bson::BinaryData

  A POD that represents a BSON element of type binary data.

Namespaces

- namespace uma
- namespace uma::bson

  Classes that present a DOM style view of a BSON document.

Functions

- UMA_BSON_API std::ofstream & uma::bson::operator<< (std::ofstream &os, const BinaryData &element)

  Writes the contents of the binary data to the specified file output stream.
- std::ostream & uma::bson::operator<< (std::ostream &os, const BinaryData::DataType &type)

  Writes the contents of the binary data to the specified output stream.
- bool uma::bson::operator== (const BinaryData &lhs, const BinaryData &rhs)

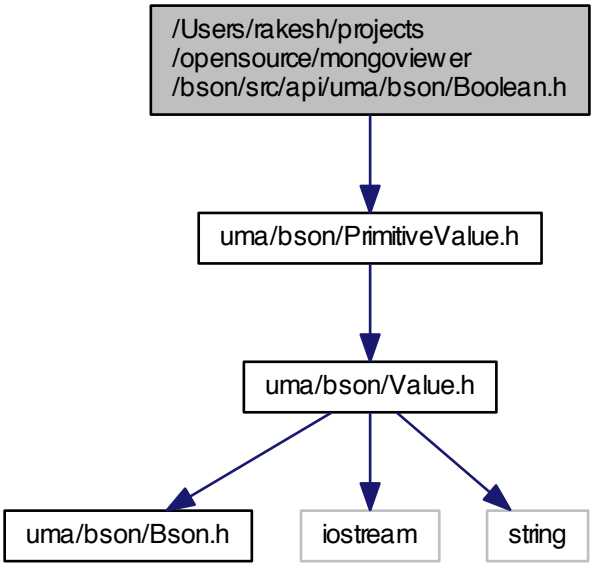
  Compare two binary data instances for equality. Compares the data stored in each instance and their types for equality.
- bool uma::bson::operator!= (const BinaryData &lhs, const BinaryData &rhs)

  Just the reverse of operator ==.

8.13 /Users/rakesh/projects/opensource/mongoviewer/bson/src/api/uma/bson/Boolean.h File Reference

```
#include <uma/bson/PrimitiveValue.h>
```

Include dependency graph for Boolean.h:



Classes

- class uma::bson::Boolean

  A POD that represents a boolean type BSON element value.

Namespaces

- namespace uma
- namespace uma::bson

  Classes that present a DOM style view of a BSON document.

8.14 /Users/rakesh/projects/opensource/mongoviewer/bson/src/api/uma/bson/Bson.h File Reference

This graph shows which files directly or indirectly include this file:

Namespaces

- namespace uma
- namespace uma::bson

  Classes that present a DOM style view of a BSON document.

Macros

- #define UMA_BSON_API

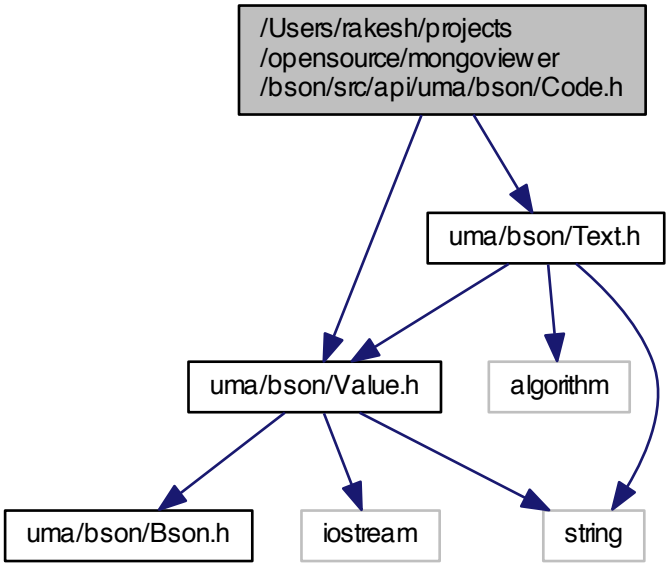
8.14.1 Macro Definition Documentation

8.14.1.1 #define UMA_BSON_API

8.15 /Users/rakesh/projects/opensource/mongoviewer/bson/src/api/uma/bson/Code.h File Reference

```
#include <uma/bson/Text.h>
#include <uma/bson/Value.h>
```

Include dependency graph for Code.h:



Classes

- class uma::bson::Code

  A POD that represents a code type BSON element value.

Namespaces

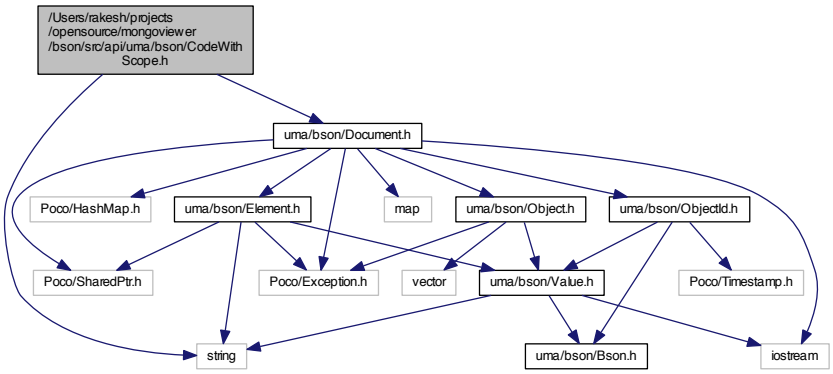
- namespace uma
- namespace uma::bson

  Classes that present a DOM style view of a BSON document.

8.16 /Users/rakesh/projects/opensource/mongoviewer/bson/src/api/uma/bson/CodeWithScope.h File Reference

```
#include <string>
#include <uma/bson/Document.h>
```

Include dependency graph for CodeWithScope.h:



Classes

- class uma::bson::CodeWithScope

  A POD that represents a code with document scope BSON element value.

Namespaces

- namespace uma
- namespace uma::bson

  Classes that present a DOM style view of a BSON document.

Functions

- bool uma::bson::operator== (const CodeWithScope &lhs, const CodeWithScope &rhs)

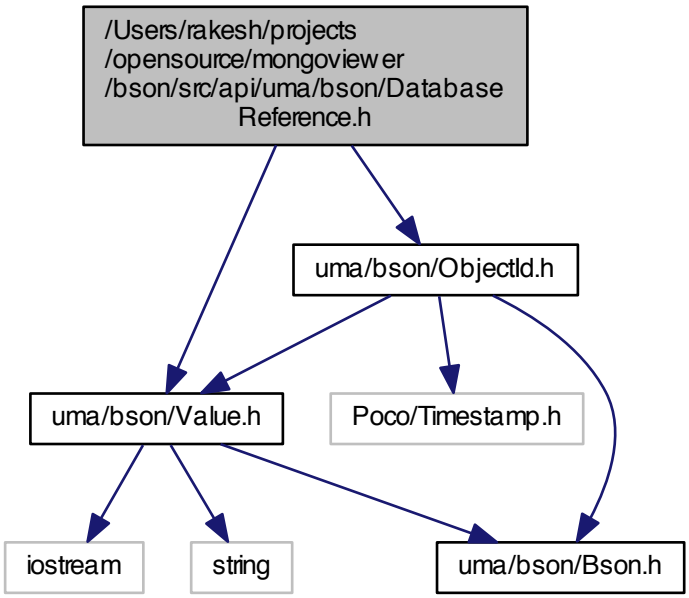
  Compare two code with scope instances for equality. Compares the code and scope values of each instance.
- bool uma::bson::operator!= (const CodeWithScope &lhs, const CodeWithScope &rhs)

  Just the reverse of operator ==.

8.17 /Users/rakesh/projects/opensource/mongoviewer/bson/src/api/uma/bson/DatabaseReference.h File Reference

```
#include <uma/bson/Value.h>
#include <uma/bson/ObjectId.h>
```

Include dependency graph for DatabaseReference.h:



Classes

- class uma::bson::DatabaseReference

  A POD that represents a database reference type BSON element value. ∗.

Namespaces

- namespace uma
- namespace uma::bson

  Classes that present a DOM style view of a BSON document.

Functions

- bool uma::bson::operator== (const DatabaseReference &lhs, const DatabaseReference &rhs)

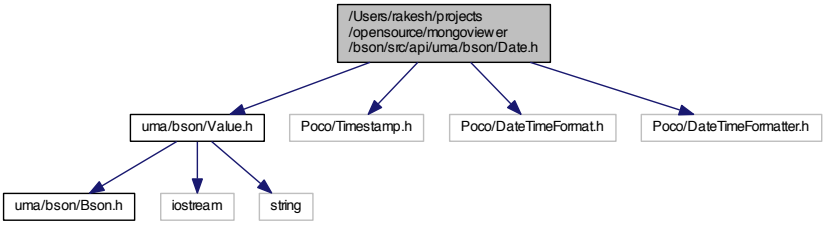
  Compare two database references for equality. Compares the names of the collections and the object id values.
- bool uma::bson::operator!= (const DatabaseReference &lhs, const DatabaseReference &rhs)

  Just the reverse of operator ==.

8.18 /Users/rakesh/projects/opensource/mongoviewer/bson/src/api/uma/bson/Date.h File Reference

```
#include <uma/bson/Value.h>
#include <Poco/Timestamp.h>
#include <Poco/DateTimeFormat.h>
#include <Poco/DateTimeFormatter.h>
```

Include dependency graph for Date.h:



Classes

- class uma::bson::Date

  A POD that represents a BSON element value of type date.

Namespaces

- namespace uma
- namespace uma::bson

  Classes that present a DOM style view of a BSON document.

Functions

- bool uma::bson::operator< (const Date &lhs, const Date &rhs)

  Compare the two data values for `less-than` operator.
- bool uma::bson::operator> (const Date &lhs, const Date &rhs)

  Compare the two data values for `greater-than` operator.
- bool uma::bson::operator== (const Date &lhs, const Date &rhs)

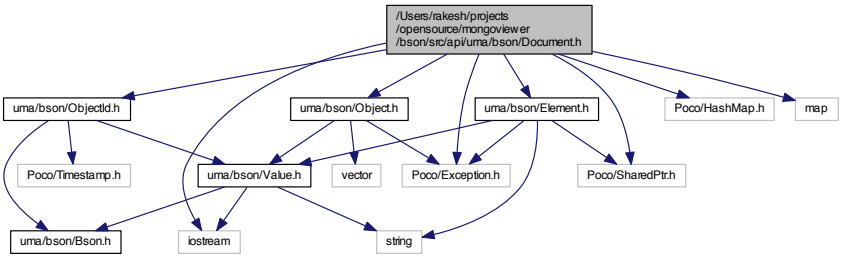
  Compare two date values for equality.
- bool uma::bson::operator!= (const Date &lhs, const Date &rhs)

  Just the reverse of operator ==.

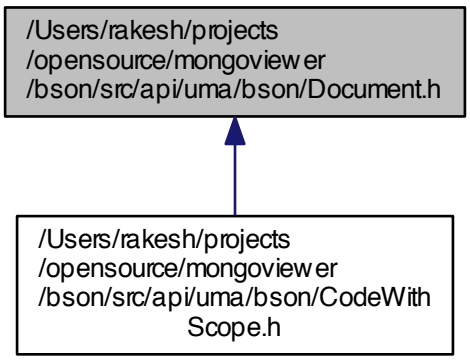
8.19 /Users/rakesh/projects/opensource/mongoviewer/bson/src/api/uma/bson/Document.h File Reference

```
#include <uma/bson/Object.h>
#include <uma/bson/Element.h>
#include <uma/bson/ObjectId.h>
#include <Poco/Exception.h>
#include <Poco/HashMap.h>
#include <Poco/SharedPtr.h>
#include <iostream>
#include <map>
```

Include dependency graph for Document.h:



This graph shows which files directly or indirectly include this file:



Classes

- class uma::bson::Document

  A class that represents a BSON Object.

Namespaces

- namespace uma

- namespace uma::bson

  Classes that present a DOM style view of a BSON document.

Functions

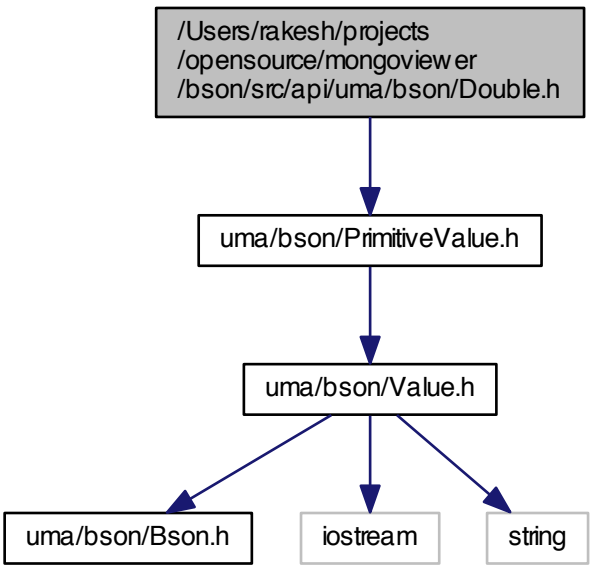
- UMA_BSON_API bool uma::bson::operator== (const Document &lhs, const Document &rhs)

  Compare two documents for equality.
- bool uma::bson::operator!= (const Document &lhs, const Document &rhs)

  Just the reverse of operator ==.

8.20 /Users/rakesh/projects/opensource/mongoviewer/bson/src/api/uma/bson/Double.h File Reference

`#include <uma/bson/PrimitiveValue.h>`
Include dependency graph for Double.h:



Classes

- class uma::bson::Double

  A POD that represents a double value in a BSON element.

Namespaces

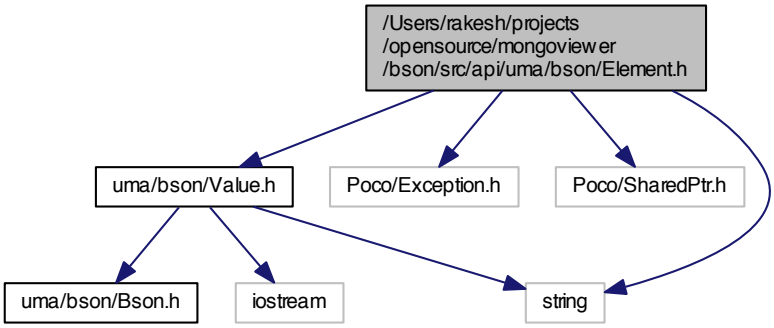
- namespace uma
- namespace uma::bson

  Classes that present a DOM style view of a BSON document.

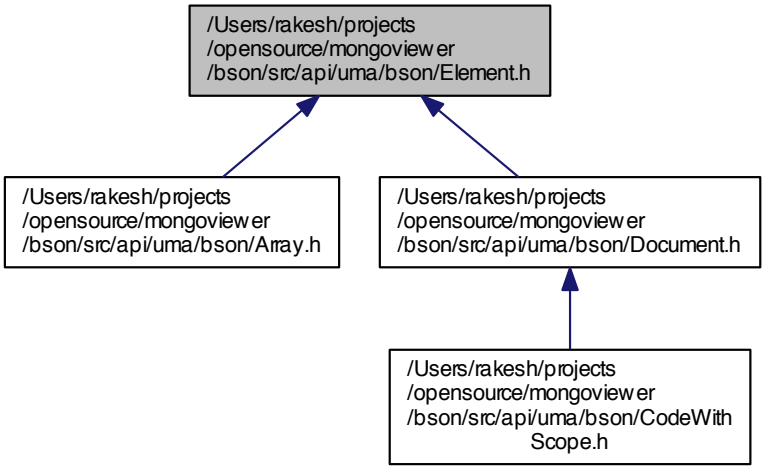
8.21 /Users/rakesh/projects/opensource/mongoviewer/bson/src/api/uma/bson/Element.h File Reference

```
#include <uma/bson/Value.h>
#include <Poco/Exception.h>
#include <Poco/SharedPtr.h>
#include <string>
```

Include dependency graph for Element.h:



This graph shows which files directly or indirectly include this file:



Classes

- class uma::bson::Element

  A class that represents a BSON element. Stores the name-value mapping for a BSON element in a PIMPL. This enables efficient copy-by-value semantics at the cost of shared data across the various copies of the element.

Namespaces

- namespace uma
- namespace uma::bson

  Classes that present a DOM style view of a BSON document.

Functions

- template<>
  UMA_BSON_API std::string uma::bson::Element::getSimple< std::string > () const

  Specialised method for getting the value of uma::bson::String.
- template<>
  UMA_BSON_API Element & uma::bson::Element::setValue< std::string > (const std::string &v)

  Specialised method for setting the value to uma::bson::String.
- bool uma::bson::operator< (const Element &lhs, const Element &rhs)

  Comparison operator for ordering element instances.
- UMA_BSON_API bool uma::bson::operator== (const Element &lhs, const Element &rhs)

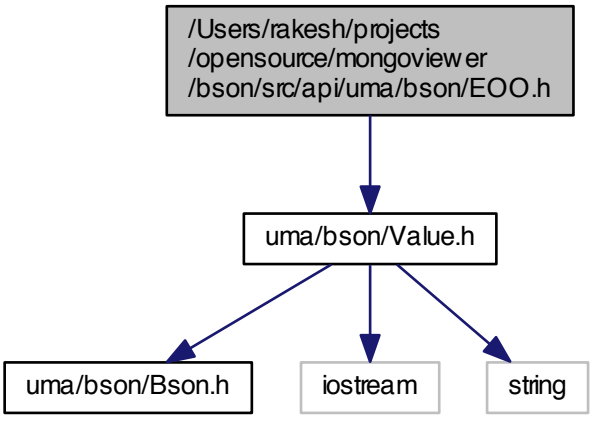
  Compare two elements for equality. Compares the names and held values of the elements.
- bool uma::bson::operator!= (const Element &lhs, const Element &rhs)

  Just the reverse of operator ==.

8.22 /Users/rakesh/projects/opensource/mongoviewer/bson/src/api/uma/bson/EOO.h File Reference

```
#include <uma/bson/Value.h>
```

Include dependency graph for EOO.h:



Classes

- class uma::bson::EOO

  A POD that represents a EOO value.

Namespaces

- namespace uma
- namespace uma::bson

  Classes that present a DOM style view of a BSON document.

Functions

- bool uma::bson::operator== (const EOO &, const EOO &)

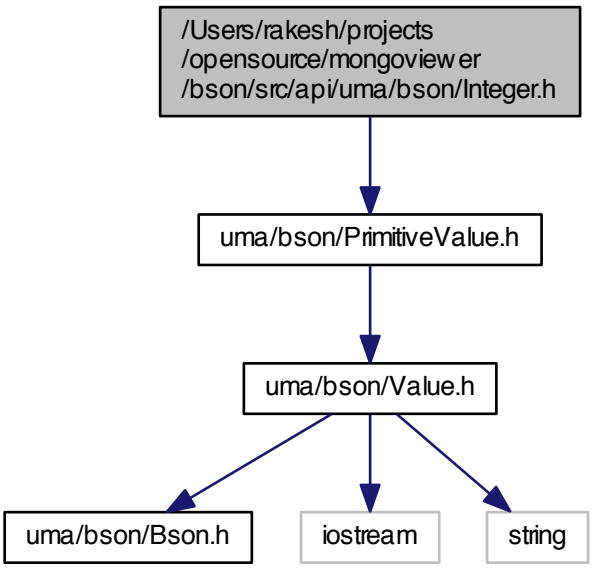
  Compare two EOO values for equality. Always returns `true`.
- bool uma::bson::operator!= (const EOO &lhs, const EOO &rhs)

  Just the reverse of operator ==.

8.23 /Users/rakesh/projects/opensource/mongoviewer/bson/src/api/uma/bson/Integer.h File Reference

```
#include <uma/bson/PrimitiveValue.h>
```

Include dependency graph for Integer.h:



Classes

- class uma::bson::Integer

  A POD that represents an integer type BSON element value.

Namespaces

- namespace uma

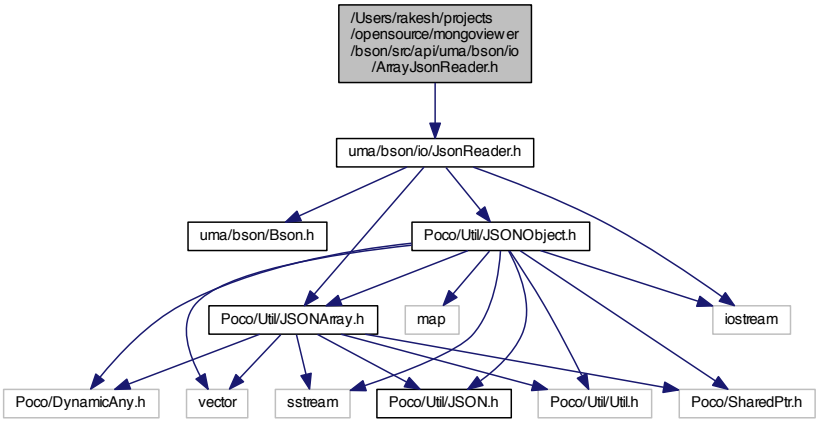
- namespace uma::bson

  Classes that present a DOM style view of a BSON document.

8.24 /Users/rakesh/projects/opensource/mongoviewer/bson/src/api/uma/bson/io/ArrayJsonReader.h File Reference

```
#include <uma/bson/io/JsonReader.h>
```

Include dependency graph for ArrayJsonReader.h:



Classes

- class uma::bson::io::ArrayJsonReader

  A parser for transforming a JSON serialised BSON array into an uma::bson::Array object representation.

Namespaces

- namespace uma
- namespace uma::bson

  Classes that present a DOM style view of a BSON document.

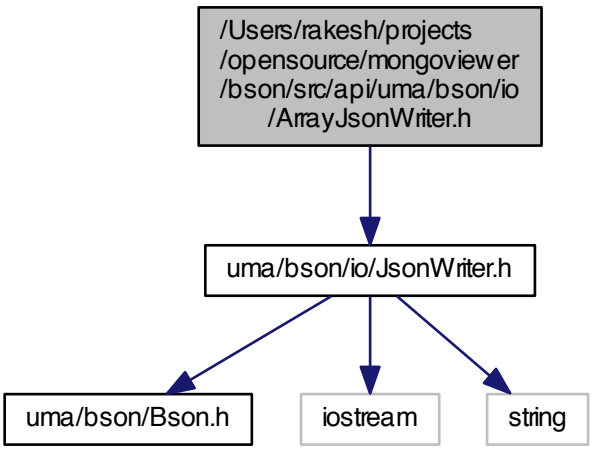
- namespace uma::bson::io

  Classes that provide IO support for BSON data.

8.25 /Users/rakesh/projects/opensource/mongoviewer/bson/src/api/uma/bson/io/ArrayJsonWriter.h File Reference

```
#include <uma/bson/io/JsonWriter.h>
```

Include dependency graph for ArrayJsonWriter.h:



Classes

- class uma::bson::io::ArrayJsonWriter

  A streaming writer that transforms a uma::bson::Array to a JSON representation.

Namespaces

- namespace uma
- namespace uma::bson

  Classes that present a DOM style view of a BSON document.

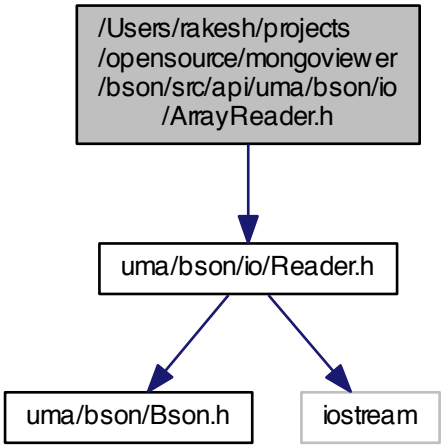
- namespace uma::bson::io

  Classes that provide IO support for BSON data.

8.26 /Users/rakesh/projects/opensource/mongoviewer/bson/src/api/uma/bson/io/ArrayReader.h File Reference

```
#include <uma/bson/io/Reader.h>
```

Include dependency graph for ArrayReader.h:



Classes

- class uma::bson::io::ArrayReader

  A streaming parser that parses a stream of BSON bytes from an input stream and transforms into a uma::bson::Array model.

Namespaces

- namespace uma
- namespace uma::bson

  Classes that present a DOM style view of a BSON document.

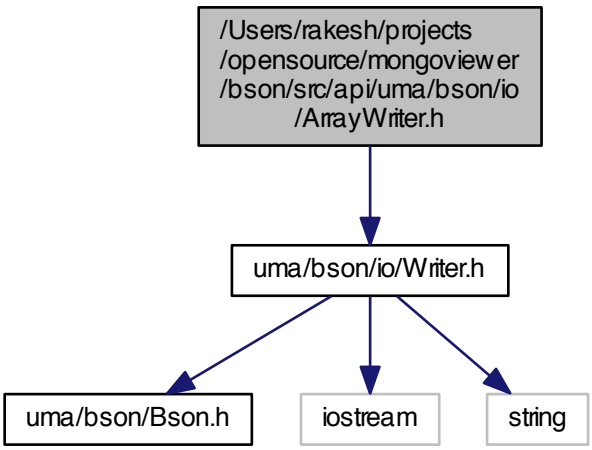
- namespace uma::bson::io

  Classes that provide IO support for BSON data.

8.27 /Users/rakesh/projects/opensource/mongoviewer/bson/src/api/uma/bson/io/ArrayWriter.h File Reference

```
#include <uma/bson/io/Writer.h>
```

Include dependency graph for ArrayWriter.h:



Classes

- class uma::bson::io::ArrayWriter

  A serialiser that writes the BSON representation of an array to a output stream.

Namespaces

- namespace uma
- namespace uma::bson

  Classes that present a DOM style view of a BSON document.

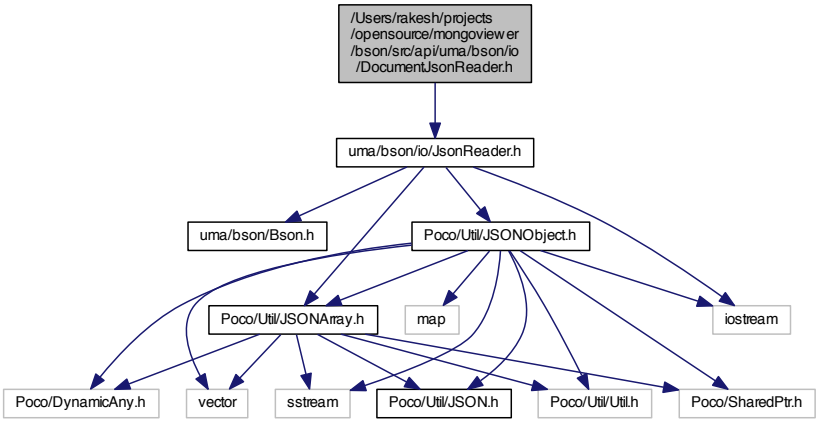
- namespace uma::bson::io

  Classes that provide IO support for BSON data.

8.28 /Users/rakesh/projects/opensource/mongoviewer/bson/src/api/uma/bson/io/DocumentJsonReader.h File Reference

```
#include <uma/bson/io/JsonReader.h>
```

Include dependency graph for DocumentJsonReader.h:



Classes

- class uma::bson::io::DocumentJsonReader

  A parser that transforms a JSON stream into a uma::bson::Document.

Namespaces

- namespace uma
- namespace uma::bson

  Classes that present a DOM style view of a BSON document.

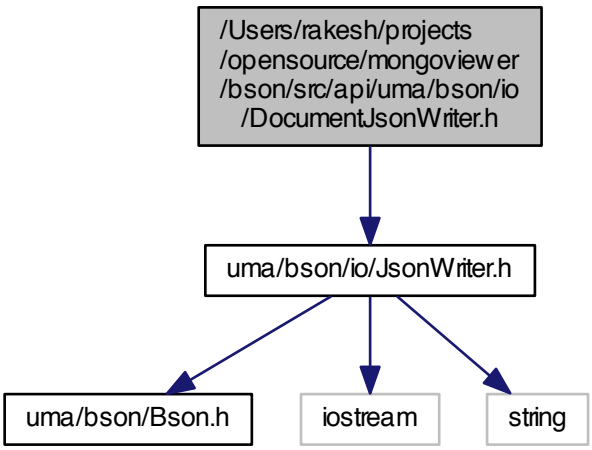
- namespace uma::bson::io

  Classes that provide IO support for BSON data.

8.29 /Users/rakesh/projects/opensource/mongoviewer/bson/src/api/uma/bson/io/DocumentJsonWriter.h File Reference

```
#include <uma/bson/io/JsonWriter.h>
```

Include dependency graph for DocumentJsonWriter.h:



Classes

- class uma::bson::io::DocumentJsonWriter

  A streaming writer that transforms a uma::bson::Document to a JSON representation.

Namespaces

- namespace uma
- namespace uma::bson

  Classes that present a DOM style view of a BSON document.

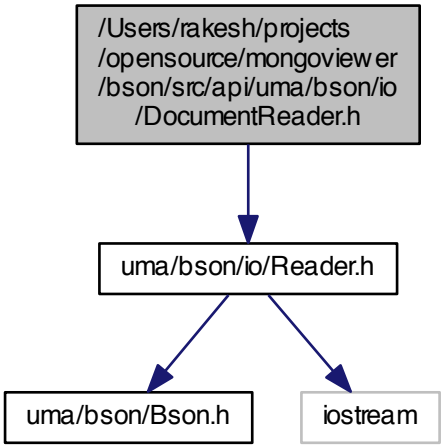
- namespace uma::bson::io

  Classes that provide IO support for BSON data.

8.30 /Users/rakesh/projects/opensource/mongoviewer/bson/src/api/uma/bson/io/DocumentReader.h File Reference

```
#include <uma/bson/io/Reader.h>
```

Include dependency graph for DocumentReader.h:



Classes

- class uma::bson::io::DocumentReader

  A streaming parser that parses a stream of BSON bytes from an input stream and transforms into a uma::bson::-Document model.

Namespaces

- namespace uma
- namespace uma::bson

  Classes that present a DOM style view of a BSON document.

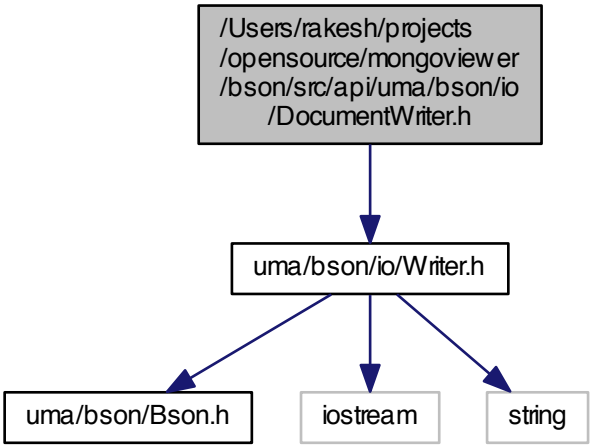
- namespace uma::bson::io

  Classes that provide IO support for BSON data.

8.31 /Users/rakesh/projects/opensource/mongoviewer/bson/src/api/uma/bson/io/Document-Writer.h File Reference

```
#include <uma/bson/io/Writer.h>
```

Include dependency graph for DocumentWriter.h:



Classes

- class uma::bson::io::DocumentWriter

  A serialiser that writes the BSON representation of a document to an output stream.

Namespaces

- namespace uma
- namespace uma::bson

  Classes that present a DOM style view of a BSON document.

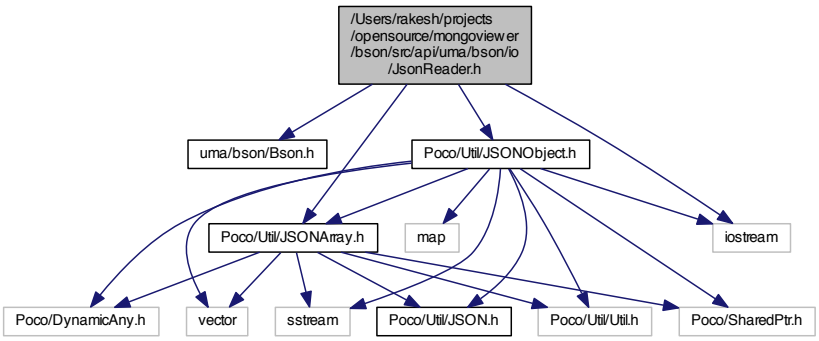
- namespace uma::bson::io

  Classes that provide IO support for BSON data.

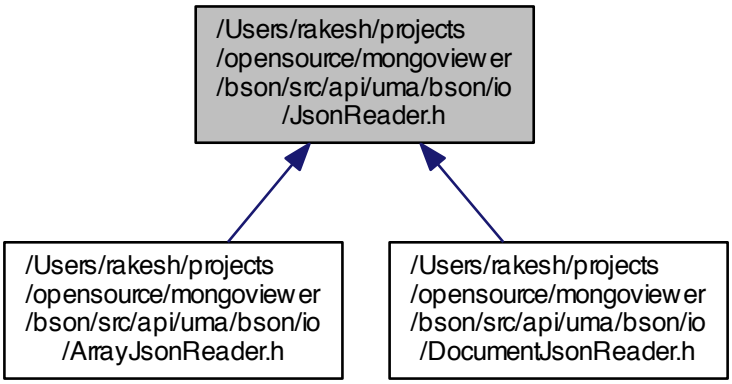
8.32 /Users/rakesh/projects/opensource/mongoviewer/bson/src/api/uma/bson/io/JsonReader.h File Reference

```
#include <uma/bson/Bson.h>
#include <Poco/Util/JSONObject.h>
#include <Poco/Util/JSONArray.h>
#include <iostream>
```

Include dependency graph for JsonReader.h:



This graph shows which files directly or indirectly include this file:



Classes

- class uma::bson::io::JsonReader

  A base streaming parser that parses a stream of JSON bytes from an input stream and transforms into a object model.

Namespaces

- namespace uma
- namespace uma::bson

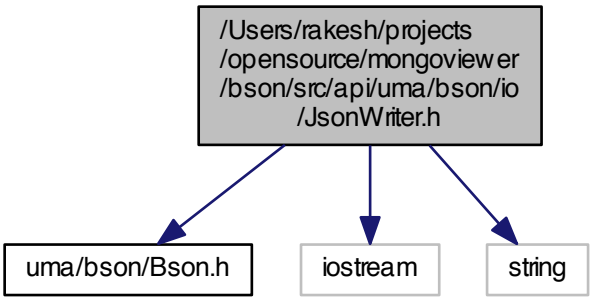
  Classes that present a DOM style view of a BSON document.
- namespace uma::bson::io

  Classes that provide IO support for BSON data.

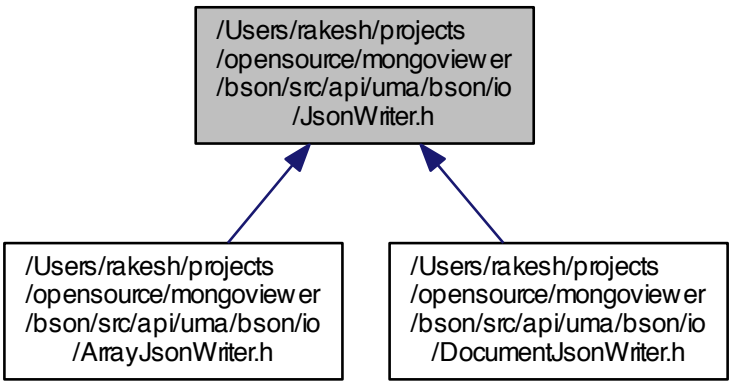
8.33 /Users/rakesh/projects/opensource/mongoviewer/bson/src/api/uma/bson/io/JsonWriter.h File Reference

```
#include <uma/bson/Bson.h>
#include <iostream>
#include <string>
```

Include dependency graph for JsonWriter.h:



This graph shows which files directly or indirectly include this file:



Classes

- class uma::bson::io::JsonWriter

  A streaming writer that transforms an object model into a JSON representation.

Namespaces

- namespace uma
- namespace uma::bson

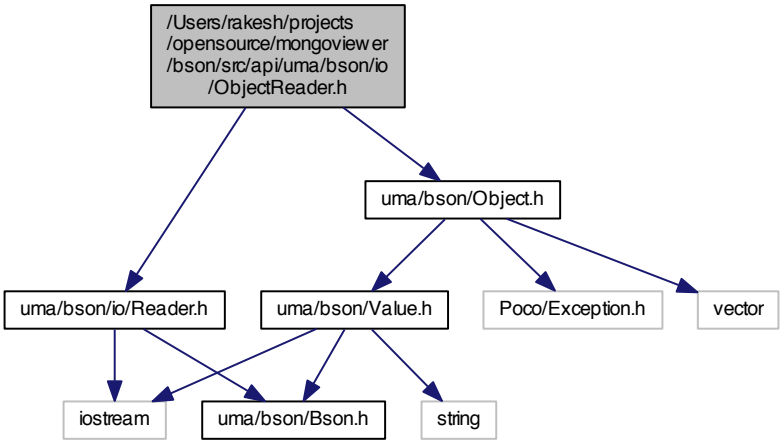
  Classes that present a DOM style view of a BSON document.
- namespace uma::bson::io

  Classes that provide IO support for BSON data.

8.34 /Users/rakesh/projects/opensource/mongoviewer/bson/src/api/uma/bson/io/ObjectReader.h File Reference

```
#include <uma/bson/io/Reader.h>
#include <uma/bson/Object.h>
```

Include dependency graph for ObjectReader.h:



Classes

- class uma::bson::io::ObjectReader

Namespaces

- namespace uma
- namespace uma::bson

  Classes that present a DOM style view of a BSON document.
- namespace uma::bson::io

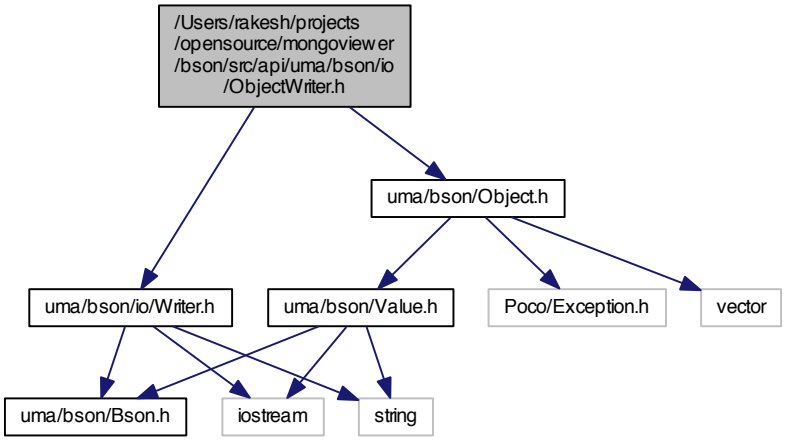
  Classes that provide IO support for BSON data.

8.35 /Users/rakesh/projects/opensource/mongoviewer/bson/src/api/uma/bson/io/ObjectWriter.h File Reference

```
#include <uma/bson/io/Writer.h>
```

```
#include <uma/bson/Object.h>
```

Include dependency graph for ObjectWriter.h:



Classes

- class uma::bson::io::ObjectWriter

Namespaces

- namespace uma
- namespace uma::bson

  Classes that present a DOM style view of a BSON document.

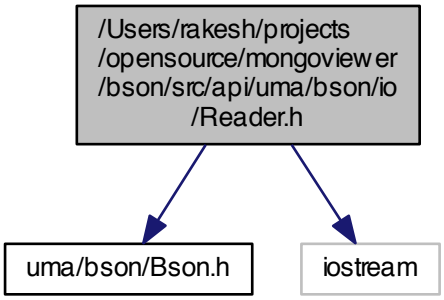
- namespace uma::bson::io

  Classes that provide IO support for BSON data.

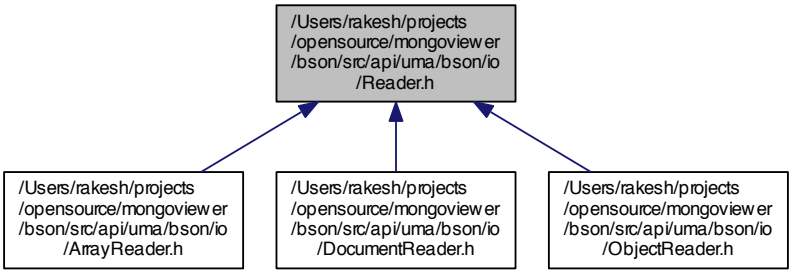
8.36 /Users/rakesh/projects/opensource/mongoviewer/bson/src/api/uma/bson/io/Reader.h File Reference

```
#include <uma/bson/Bson.h>
#include <iostream>
```

Include dependency graph for Reader.h:



This graph shows which files directly or indirectly include this file:



Classes

- class uma::bson::io::Reader

  A base streaming parser that parses a stream of BSON bytes from an input stream and transforms into a object model.

Namespaces

- namespace uma
- namespace uma::bson

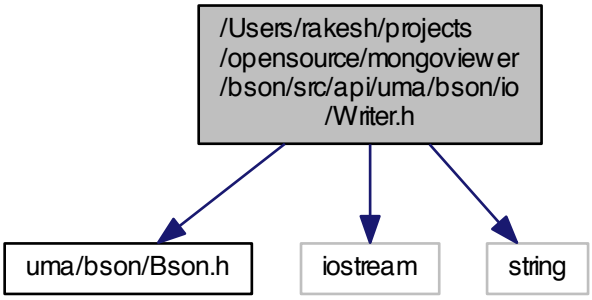
  Classes that present a DOM style view of a BSON document.
- namespace uma::bson::io

  Classes that provide IO support for BSON data.

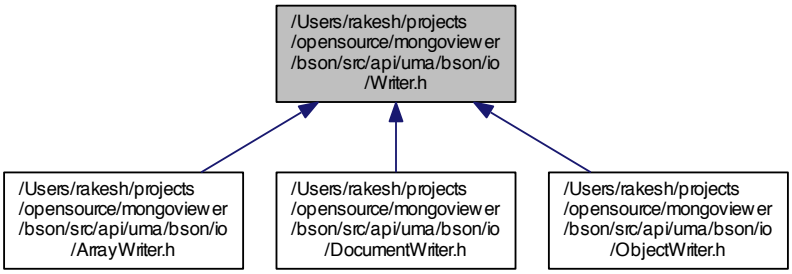
8.37 /Users/rakesh/projects/opensource/mongoviewer/bson/src/api/uma/bson/io/Writer.h File Reference

```
#include <uma/bson/Bson.h>
#include <iostream>
#include <string>
```

Include dependency graph for Writer.h:



This graph shows which files directly or indirectly include this file:



Classes

- class uma::bson::io::Writer

Namespaces

- namespace uma
- namespace uma::bson

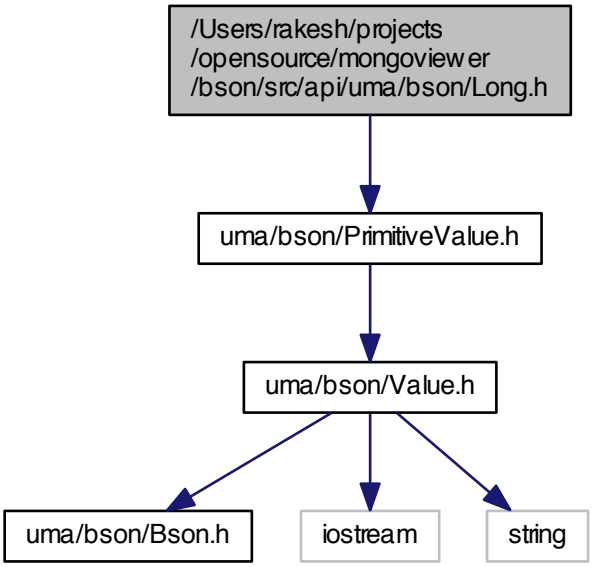
  Classes that present a DOM style view of a BSON document.
- namespace uma::bson::io

  Classes that provide IO support for BSON data.

8.38 /Users/rakesh/projects/opensource/mongoviewer/bson/src/api/uma/bson/Long.h File Reference

```
#include <uma/bson/PrimitiveValue.h>
```

Include dependency graph for Long.h:



Classes

- class uma::bson::Long

  A POD that represents a long (64 bit integer) type BSON element value.

Namespaces

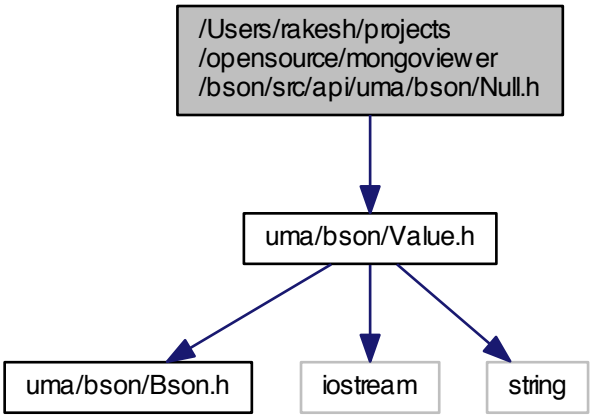
- namespace uma
- namespace uma::bson

  Classes that present a DOM style view of a BSON document.

8.39 /Users/rakesh/projects/opensource/mongoviewer/bson/src/api/uma/bson/Null.h File Reference

```
#include <uma/bson/Value.h>
```

Include dependency graph for Null.h:



Classes

- class uma::bson::Null

  A POD that represents a null BSON element value.

Namespaces

- namespace uma
- namespace uma::bson

  Classes that present a DOM style view of a BSON document.

Functions

- bool uma::bson::operator== (const Null &, const Null &)

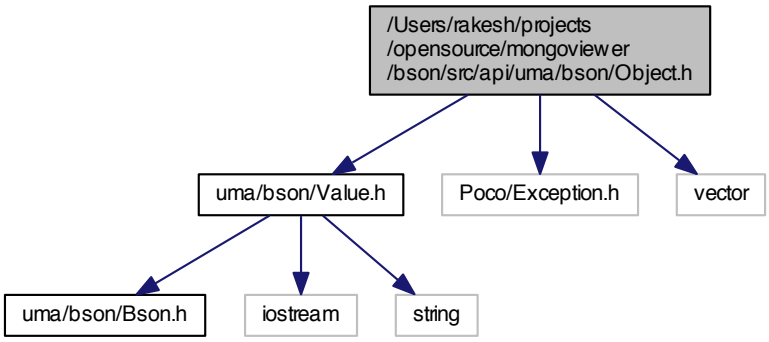
  Compare two null values for equality. Always returns `true`.
- bool uma::bson::operator!= (const Null &lhs, const Null &rhs)

  Just the reverse of operator ==.

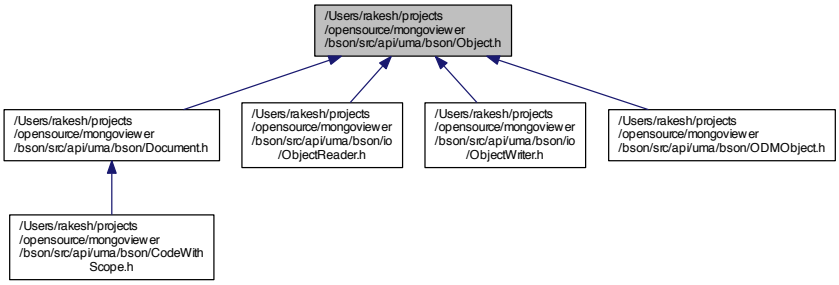
8.40 /Users/rakesh/projects/opensource/mongoviewer/bson/src/api/uma/bson/Object.h File Reference

```
#include <uma/bson/Value.h>
#include <Poco/Exception.h>
#include <vector>
```

Include dependency graph for Object.h:



This graph shows which files directly or indirectly include this file:



Classes

- class uma::bson::Object

  Abstract base class that represents a BSON Object type.

Namespaces

- namespace uma
- namespace uma::bson

  Classes that present a DOM style view of a BSON document.

Functions

- UMA_BSON_API bool uma::bson::operator== (const Object &lhs, const Object &rhs)

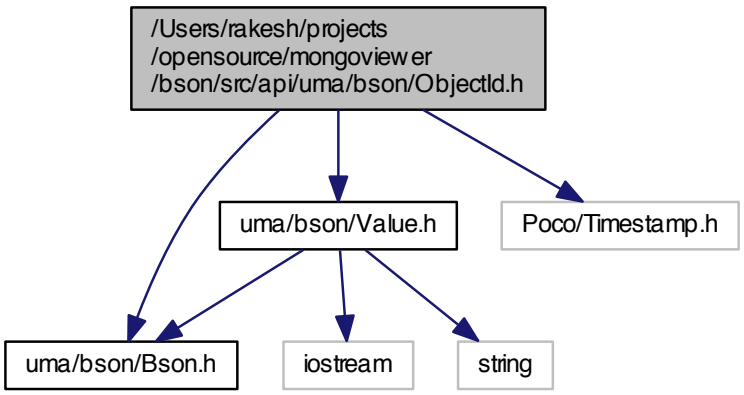
  Compare two objects for equality.
- bool uma::bson::operator!= (const Object &lhs, const Object &rhs)

  Just the reverse of operator ==.

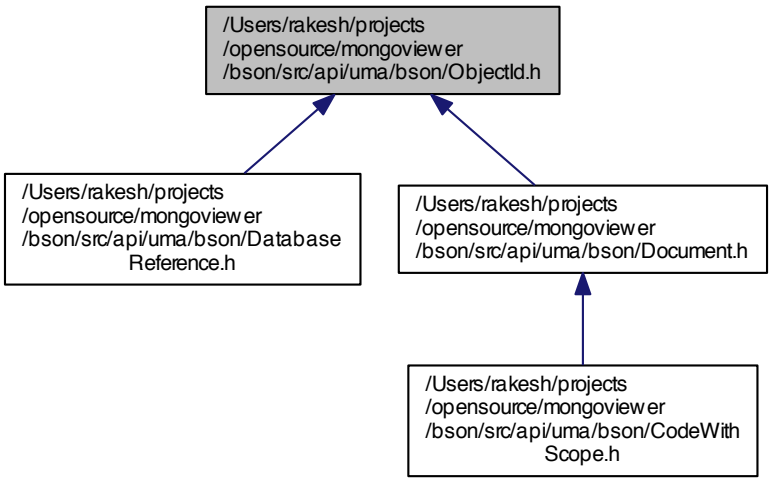
8.41 /Users/rakesh/projects/opensource/mongoviewer/bson/src/api/uma/bson/ObjectId.h File Reference

```
#include <uma/bson/Value.h>
#include <uma/bson/Bson.h>
#include <Poco/Timestamp.h>
```

Include dependency graph for ObjectId.h:



This graph shows which files directly or indirectly include this file:



Classes

- class uma::bson::ObjectId

  A POD that represents a MongoDB OID type.

Namespaces

- namespace uma
- namespace uma::bson

  Classes that present a DOM style view of a BSON document.

Functions

- bool uma::bson::operator< (const ObjectId &lhs, const ObjectId &rhs)

  Less than operator for comparing ObjectId instances.
- bool uma::bson::operator> (const ObjectId &lhs, const ObjectId &rhs)

  Greater than operator for comparing ObjectId instances.
- UMA_BSON_API bool uma::bson::operator== (const ObjectId &lhs, const ObjectId &rhs)

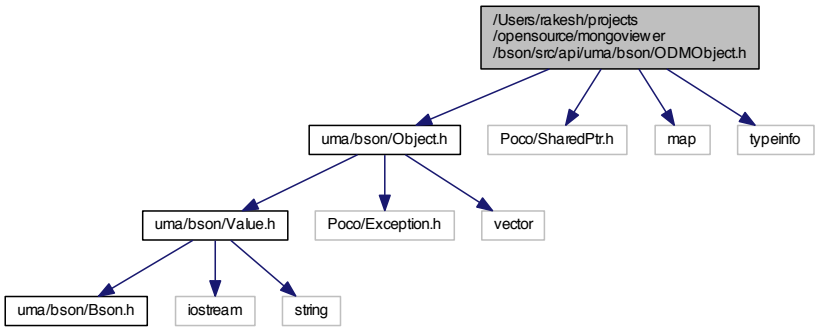
  Compare two object id values for equality. Compares the data in each instance.
- bool uma::bson::operator!= (const ObjectId &lhs, const ObjectId &rhs)

  Just the reverse of operator ==.

8.42 /Users/rakesh/projects/opensource/mongoviewer/bson/src/api/uma/bson/ODMObject.h File Reference

```
#include <uma/bson/Object.h>
#include <Poco/SharedPtr.h>
#include <map>
#include <typeinfo>
```

Include dependency graph for ODMObject.h:



Classes

- class uma::bson::ODMObject< Model >

  Abstract class that presents a more friendly ODM interface than uma::bson::Object.
- class uma::bson::ODMObject< Model >::MetaField

  Abstract base class that encapsulates a field in a model object. Primarily used to get around requirement that implementation needs exact type of data encapsulated in the field as template type.
- class uma::bson::ODMObject< Model >::MetaFieldImpl< DataType >

  Encapsulates a field in a model object. Fields are represented as a triplet of the field in the class, and its accessor and mutator methods.

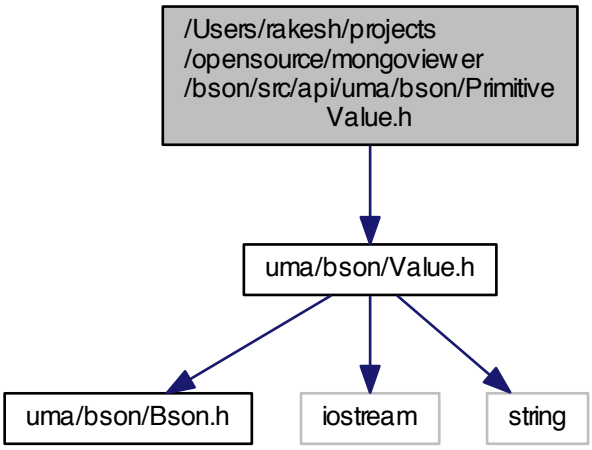
Namespaces

- namespace uma
- namespace uma::bson

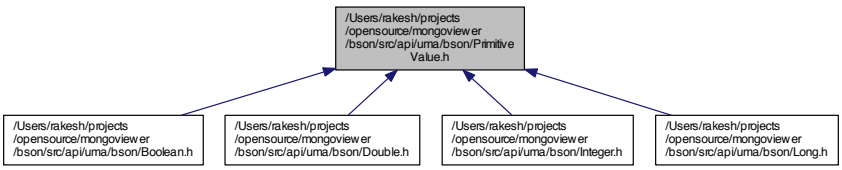
  Classes that present a DOM style view of a BSON document.

8.43 /Users/rakesh/projects/opensource/mongoviewer/bson/src/api/uma/bson/PrimitiveValue.h File Reference

`#include <uma/bson/Value.h>`
Include dependency graph for PrimitiveValue.h:



This graph shows which files directly or indirectly include this file:



Classes

- class uma::bson::PrimitiveValue< T >

  A base value class used to represent primitive type values.

Namespaces

- namespace uma

- namespace uma::bson

  Classes that present a DOM style view of a BSON document.

Functions

- template<typename T >
  bool uma::bson::operator== (const PrimitiveValue< T > &lhs, const PrimitiveValue< T > &rhs)

  Compare the encapsulated values in the specified instances for equality;.

- template<typename T >
  bool uma::bson::operator== (const PrimitiveValue< T > &lhs, const T rhs)

  Compare the encapsulated value in the specified instance for equality with the specified primitive value.

- template<typename T >
  bool uma::bson::operator!= (const PrimitiveValue< T > &lhs, const PrimitiveValue< T > &rhs)

  Just the reverse of operator ==.

- template<typename T >
  bool uma::bson::operator!= (const PrimitiveValue< T > &lhs, const T rhs)

  Just the reverse of operator ==.

- template<typename T >
  bool uma::bson::operator< (const PrimitiveValue< T > &lhs, const PrimitiveValue< T > &rhs)

  Check to see if the value encapsulated in the left hand instance is less than the value in the right hand instance.

- template<typename T >
  bool uma::bson::operator< (const PrimitiveValue< T > &lhs, const T rhs)

  Compare the encapsulated value in the left hand instance with the primitive value specified.

- template<typename T >
  bool uma::bson::operator> (const PrimitiveValue< T > &lhs, const PrimitiveValue< T > &rhs)

  Compare the encapsulated values of the two instances.

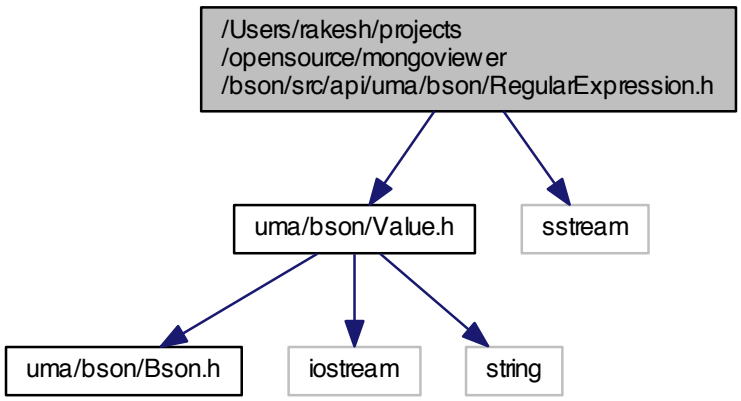
- template<typename T >
  bool uma::bson::operator> (const PrimitiveValue< T > &lhs, const T rhs)

  Compare the encapsulated value in the left hand instance with the primitive value in the right hand instance.

8.44 /Users/rakesh/projects/opensource/mongoviewer/bson/src/api/uma/bson/RegularExpression.h File Reference

```
#include <uma/bson/Value.h>
#include <sstream>
```

Include dependency graph for RegularExpression.h:



Classes

- class uma::bson::RegularExpression

  A POD that represents a regular expression type BSON element value.

Namespaces

- namespace uma
- namespace uma::bson

  Classes that present a DOM style view of a BSON document.

Functions

- bool uma::bson::operator== (const RegularExpression &lhs, const RegularExpression &rhs)

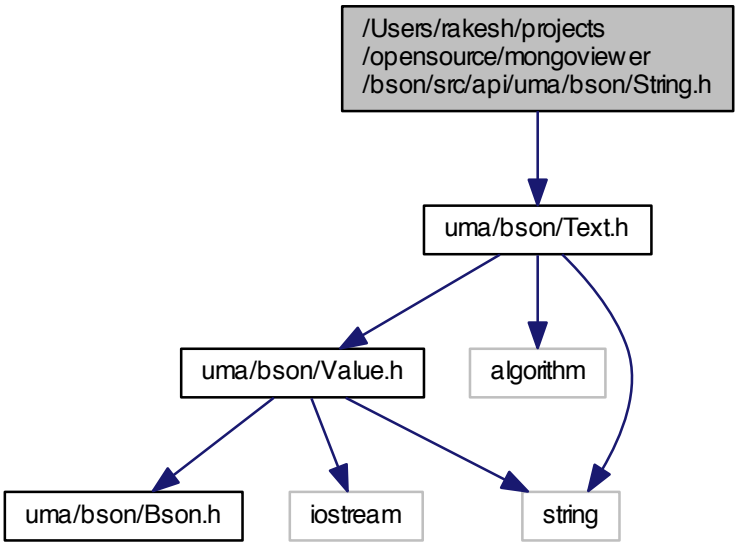
  Compare two regular expressions for equality. Compares the `regex` and `flags` values for equality.
- bool uma::bson::operator!= (const RegularExpression &lhs, const RegularExpression &rhs)

  Just the reverse of operator ==.

8.45 /Users/rakesh/projects/opensource/mongoviewer/bson/src/api/uma/bson/String.h File Reference

```
#include <uma/bson/Text.h>
```

Include dependency graph for String.h:



Classes

- class uma::bson::String

  A POD that represents a string value in a BSON element.

Namespaces

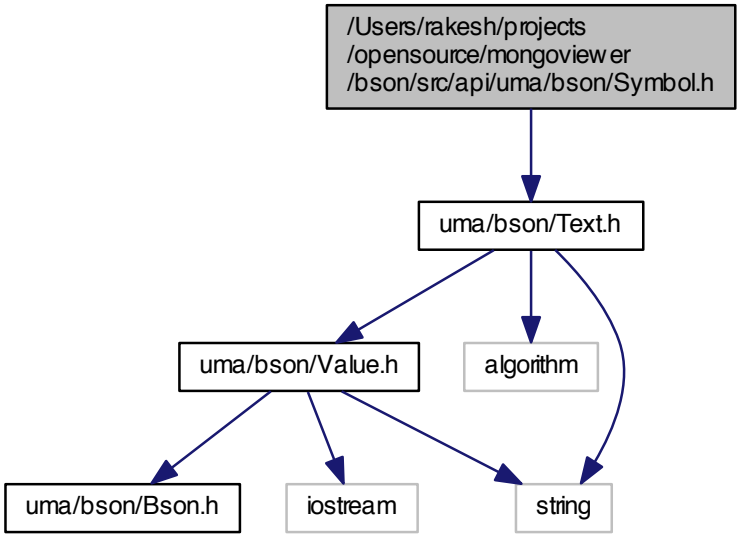
- namespace uma
- namespace uma::bson

  Classes that present a DOM style view of a BSON document.

8.46 /Users/rakesh/projects/opensource/mongoviewer/bson/src/api/uma/bson/Symbol.h File Reference

```
#include <uma/bson/Text.h>
```

Include dependency graph for Symbol.h:



Classes

- class uma::bson::Symbol

  A POD that represents a programming language symbol type BSON element value.

Namespaces

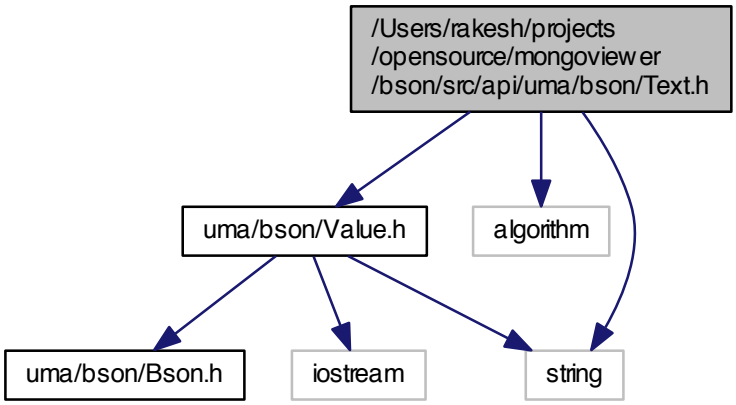
- namespace uma
- namespace uma::bson

  Classes that present a DOM style view of a BSON document.

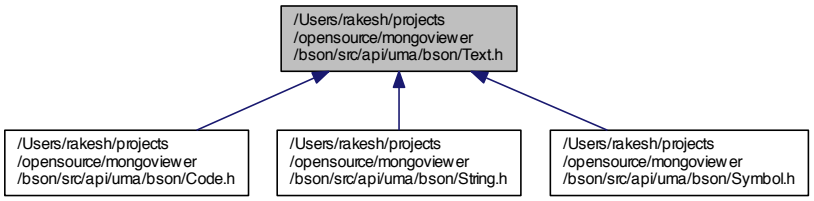
8.47 /Users/rakesh/projects/opensource/mongoviewer/bson/src/api/uma/bson/Text.h File Reference

```
#include <uma/bson/Value.h>
#include <algorithm>
#include <string>
```

Include dependency graph for Text.h:



This graph shows which files directly or indirectly include this file:



Classes

- class uma::bson::Text

  A class that holds a string. BSON string/text values can be large, hence take care when copy constructing or assigning text instances.

Namespaces

- namespace uma
- namespace uma::bson

  Classes that present a DOM style view of a BSON document.

Functions

- bool uma::bson::operator== (const Text &lhs, const Text &rhs)

  Compare the two text instances for equality.
- bool uma::bson::operator== (const Text &text, const std::string &str)

  Compare the held value with the specified value.

- bool uma::bson::operator< (const Text &lhs, const Text &rhs)

  Compare the two text instances. Compares the held values of both instances.
- bool uma::bson::operator< (const Text &text, const std::string &str)

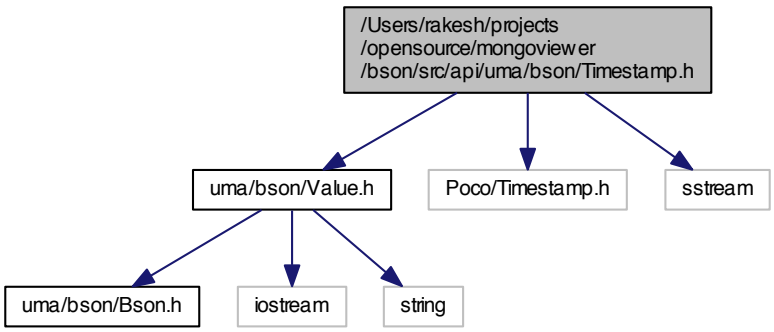
  Compares the value held in the specified text with the specified string value.
- bool uma::bson::operator!= (const Text &lhs, const Text &rhs)

  Just the reverse of operator ==.

8.48 /Users/rakesh/projects/opensource/mongoviewer/bson/src/api/uma/bson/Timestamp.h File Reference

```
#include <uma/bson/Value.h>
#include <Poco/Timestamp.h>
#include <sstream>
```

Include dependency graph for Timestamp.h:



Classes

- class uma::bson::Timestamp

  A POD that represents a database timestamp BSON element value.

Namespaces

- namespace uma
- namespace uma::bson

  Classes that present a DOM style view of a BSON document.

Functions

- bool uma::bson::operator== (const Timestamp &lhs, const Timestamp &rhs)

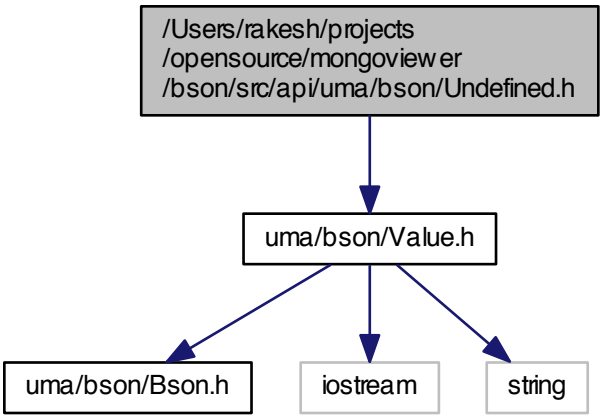
  Compare two timestamps for equality. Compares the timestamp and increment values of both instances for equality.
- bool uma::bson::operator!= (const Timestamp &lhs, const Timestamp &rhs)

  Just the reverse of operator ==.

8.49 /Users/rakesh/projects/opensource/mongoviewer/bson/src/api/uma/bson/Undefined.h File Reference

```
#include <uma/bson/Value.h>
```

Include dependency graph for Undefined.h:



Classes

- class uma::bson::Undefined

Namespaces

- namespace uma
- namespace uma::bson

  Classes that present a DOM style view of a BSON document.

Functions

- bool uma::bson::operator== (const Undefined &, const Undefined &)

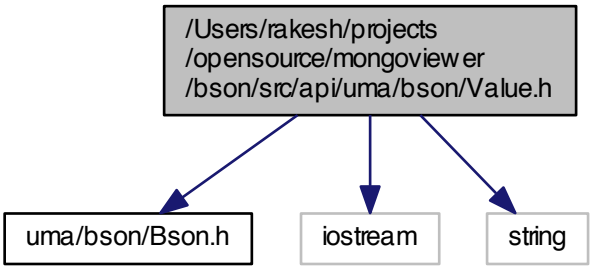
  Compare two undefined values for equality. Always returns true.
- bool uma::bson::operator!= (const Undefined &lhs, const Undefined &rhs)

  Just the reverse of operator ==.

8.50 /Users/rakesh/projects/opensource/mongoviewer/bson/src/api/uma/bson/Value.h File Reference

```
#include <uma/bson/Bson.h>
#include <iostream>
#include <string>
```

Include dependency graph for Value.h:



This graph shows which files directly or indirectly include this file:

Classes

- class uma::bson::Value

Namespaces

- namespace uma
- namespace uma::bson

  Classes that present a DOM style view of a BSON document.

Functions

- std::ostream & uma::bson::operator<< (std::ostream &os, const Value::Type &type)

  Write integer representation of the value type to the specified output stream. This is primarily intended for use while debugging/logging. Note that this cannot be used to write to BSON, since types are written as single byte char values.
- UMA_BSON_API bool uma::bson::operator== (const Value &lhs, const Value &rhs)

  Compare two values for equality. Default implementation performs a true comparison of the standard value type implementations provided by this API.
- bool uma::bson::operator!= (const Value &lhs, const Value &rhs)

  Just the reverse of operator ==.

Index