LiteDB

Fork me on GitHub

**Docs**

# BsonDocument

The `BsonDocument` class is LiteDB's implementation of documents. Internally, a `BsonDocument` stores key-value pairs in a `Dictionary<string, BsonValue>`.

```
var customer = new BsonDocument();
customer["_id"] = ObjectId.NewObjectId();
customer["Name"] = "John Doe";
customer["CreateDate"] = DateTime.Now;
customer["Phones"] = new BsonArray { "8000-0000", "9000-000" };
customer["IsActive"] = true;
customer["IsAdmin"] = new BsonValue(true);
customer["Address"] = new BsonDocument
{
    ["Street"] = "Av. Protasio Alves"
};
customer["Address"]["Number"] = "1331";
```

LiteDB supports documents up to 16MB after BSON serialization.

About document field **keys**:

- Keys are case-insensitive
- Duplicate keys are not allowed
- LiteDB keeps the original key order, including mapped classes. The only exception is for `_id` field, which will always be the first field.

About document field **values**:

- Values can be any BSON value data type: Null, Int32, Int64, Decimal, Double, String, Embedded Document, Array, Binary, ObjectId, Guid, Boolean, DateTime, MinValue, MaxValue
- When a field is indexed, the value must be less than 256 bytes after BSON serialization.
- `_id` field cannot be: `Null`, `MinValue` or `MaxValue`
- `_id` is unique indexed field, so value must be less than 256 bytes

About .NET classes

- `BsonValue`
  - This class can hold any BSON data type, including null, array or document.
  - Has implicit constructor to all supported .NET data types
  - Value never changes (immutable)
  - `RawValue` property that returns internal .NET object instance
- `BsonArray`
  - Supports `IEnumerable<BsonValue>`
  - Each array item can have different BSON type objects
- `BsonDocument`
  - Missing fields always return `BsonValue.Null` value

```
// Testing BSON value data type
if(customer["Name"].IsString) { ... }

// Helper to get .NET type
string str = customer["Name"].AsString;
```

To use other .NET data types you need a custom `BsonMapper` class.