

# \$project (aggregation)

## On this page

- Definition
- Considerations
- Examples

## Definition

### \$project

Passes along the documents with the requested fields to the next stage in the pipeline. The specified fields can be existing fields from the input documents or newly computed fields.

The \$project stage has the following prototype form:

```
{ $project: { <specification(s)> } }
```

The \$project takes a document that can specify the inclusion of fields, the suppression of the `_id` field, the addition of new fields, and the resetting of the values of existing fields. Alternatively, you may specify the *exclusion* of fields.

The \$project specifications have the following forms:

Form	Description
<code>&lt;field&gt;: &lt;1 or true&gt;</code>	Specify the inclusion of a field.
<code>_id: &lt;0 or false&gt;</code>	Specify the suppression of the <code>_id</code> field.
<code>&lt;field&gt;: &lt;expression&gt;</code>	Add a new field or reset the value of an existing field.
<code>&lt;field&gt;:&lt;0 or false&gt;</code>	<div><i>New in version 3.4.</i></div> <div>Specify the exclusion of a field.</div> <div>If you specify the exclusion of a field other than <code>_id</code>, you <b>cannot</b> employ any other \$project specification forms.</div>

# Considerations



## Include Existing Fields

- The `_id` field is, by default, included in the output documents. To include any other fields from the input documents in the output documents, you must explicitly specify the inclusion in `$project`.
- If you specify an inclusion of a field that does not exist in the document, `$project` ignores that field inclusion and does not add the field to the document.

## Suppress the `_id` Field

By default, the `_id` field is included in the output documents. To exclude the `_id` field from the output documents, you must explicitly specify the suppression of the `_id` field in `$project`.

## Exclude Fields

*New in version 3.4.*

If you specify the exclusion of a field or fields, all other fields are returned in the output documents.

If you specify the exclusion of a field other than `_id`, you cannot employ any other `$project` specification forms: i.e. if you exclude fields, you cannot also specify the inclusion of fields, reset the value of existing fields, or add new fields.

## Add New Fields or Reset Existing Fields

To add a new field or to reset the value of an existing field, specify the field name and set its value to some expression. For more information on expressions, see [Expressions](#).

To set a field value directly to a numeric or boolean literal, as opposed to setting the field to an expression that resolves to a literal, use the `$literal` operator. Otherwise, `$project` treats the numeric or boolean literal as a flag for including or excluding the field.

By specifying a new field and setting its value to the field path of an existing field, you can effectively rename a field.

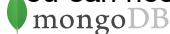
*Changed in version 3.2:* Starting in MongoDB 3.2, `$project` stage supports using the square brackets `[]` to directly create new array fields. If array specification includes fields that are non-existent in a document, the operation substitutes `null` as the value for that field. For an example, see [Project New Array Fields](#).

## Embedded Document Fields

When projecting or adding/resetting a field within an embedded document, you can either use dot notation, as in

```
"contact.address.country": <1 or 0 or expression>
```

Or you can nest the fields:



```
contact: { address: { country: <1 or 0 or expression> } }
```

When nesting the fields, you *cannot* use dot notation inside the embedded document to specify the field, e.g. `contact: { "address.country": <1 or 0 or expression> }` is *invalid*.

## Restrictions

*Changed in version 3.4.*

MongoDB 3.4 and later produces an error if the `$project` specification is an empty document.

## Examples

### Include Specific Fields in Output Documents

Consider a `books` collection with the following document:

```
{
  "_id" : 1,
  title: "abc123",
  isbn: "0001122223334",
  author: { last: "zzz", first: "aaa" },
  copies: 5
}
```

The following `$project` stage includes only the `_id`, `title`, and the `author` fields in its output documents:

```
db.books.aggregate( [ { $project : { title : 1 , author : 1 } } ] )
```

The operation results in the following document:

```
{ "_id" : 1, "title" : "abc123", "author" : { "last" : "zzz", "first" : "aaa" } }
```

### Suppress `_id` Field in the Output Documents

The `_id` field is always included by default. To exclude the `_id` field from the output documents of the `$project` stage, specify the exclusion of the `_id` field by setting it to `0` in the projection document.

Consider a `books` collection with the following document:

```
mongoDB
{
  "_id" : 1,
  title: "abc123",
  isbn: "0001122223334",
  author: { last: "zzz", first: "aaa" },
  copies: 5
}
```

The following `$project` stage excludes the `_id` field but includes the `title`, and the `author` fields in its output documents:

```
db.books.aggregate( [ { $project : { _id: 0, title : 1 , author : 1 } } ] )
```

The operation results in the following document:

```
{ "title" : "abc123", "author" : { "last" : "zzz", "first" : "aaa" } }
```

## Exclude Fields from Output Documents

*New in version 3.4.*

Consider a `books` collection with the following document:

```
{
  "_id" : 1,
  title: "abc123",
  isbn: "0001122223334",
  author: { last: "zzz", first: "aaa" },
  copies: 5,
  lastModified: "2016-07-28"
}
```

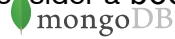
The following `$project` stage excludes the `lastModified` field from the output:

```
db.books.aggregate( [ { $project : { "lastModified": 0 } } ] )
```

## Exclude Fields from Embedded Documents

*New in version 3.4.*

Consider a `books` collection with the following document:



```
{
  "_id" : 1,
  "title": "abc123",
  "isbn": "0001122223334",
  "author": { last: "zzz", first: "aaa" },
  "copies": 5,
  "lastModified": "2016-07-28"
}
```

The following `$project` stage excludes the `author.first` and `lastModified` fields from the output:

```
db.books.aggregate( [ { $project : { "author.first" : 0, "lastModified" : 0 } } ] )
```

Alternatively, you can nest the exclusion specification in a document:

```
db.bookmarks.aggregate( [ { $project: { "author": { "first": 0}, "lastModified" : 0 } } ] )
```

Both specifications result in the same output:

```
{
  "_id" : 1,
  "title" : "abc123",
  "isbn" : "0001122223334",
  "author" : {
    "last" : "zzz"
  },
  "copies" : 5,
}
```

## Include Specific Fields from Embedded Documents

Consider a `bookmarks` collection with the following documents:

```
{ _id: 1, user: "1234", stop: { title: "book1", author: "xyz", page: 32 } }
{ _id: 2, user: "7890", stop: [ { title: "book2", author: "abc", page: 5 }, { title: "boo
```

To include only the `title` field in the embedded document in the `stop` field, you can use the dot notation:



```
db.bookmarks.aggregate( [ { $project: { "stop.title": 1 } } ] )
```

Or, you can nest the inclusion specification in a document:

```
db.bookmarks.aggregate( [ { $project: { stop: { title: 1 } } } ] )
```

Both specifications result in the following documents:


```
{ "_id" : 1, "stop" : { "title" : "book1" } }
{ "_id" : 2, "stop" : [ { "title" : "book2" }, { "title" : "book3" } ] }
```

## Include Computed Fields

Consider a `books` collection with the following document:

```
{
  "_id" : 1,
  "title": "abc123",
  "isbn": "0001122223334",
  "author": { last: "zzz", first: "aaa" },
  "copies": 5
}
```

The following `$project` stage adds the new fields `isbn`, `lastName`, and `copiesSold`:



```
db.books.aggregate([
  {
    $project: {
      title: 1,
      isbn: {
        prefix: { $substr: [ "$isbn", 0, 3 ] },
        group: { $substr: [ "$isbn", 3, 2 ] },
        publisher: { $substr: [ "$isbn", 5, 4 ] },
        title: { $substr: [ "$isbn", 9, 3 ] },
        checkDigit: { $substr: [ "$isbn", 12, 1 ] }
      },
      lastName: "$author.last",
      copiesSold: "$copies"
    }
  }
])
```

The operation results in the following document:

```
{
  "_id" : 1,
  "title" : "abc123",
  "isbn" : {
    "prefix" : "000",
    "group" : "11",
    "publisher" : "2222",
    "title" : "333",
    "checkDigit" : "4"
  },
  "lastName" : "zzz",
  "copiesSold" : 5
}
```

## Project New Array Fields

For example, if a collection includes the following document:

```
{ "_id" : ObjectId("55ad167f320c6be244eb3b95"), "x" : 1, "y" : 1 }
```

The following operation projects the fields `x` and `y` as elements in a new field `myArray`:



```
db.collection.aggregate( [ { $project: { myArray: [ "$x", "$y" ] } } ] )
```

The operation returns the following document:

```
{ "_id" : ObjectId("55ad167f320c6be244eb3b95"), "myArray" : [ 1, 1 ] }
```

If array specification includes fields that are non-existent in a document, the operation substitutes `null` as the value for that field.

For example, given the same document as above, the following operation projects the fields `x`, `y`, and a non-existing field `$someField` as elements in a new field `myArray`:

```
db.collection.aggregate( [ { $project: { myArray: [ "$x", "$y", "$someField" ] } } ] )
```

The operation returns the following document:

```
{ "_id" : ObjectId("55ad167f320c6be244eb3b95"), "myArray" : [ 1, 1, null ] }
```

#### SEE ALSO:

Aggregation with the Zip Code Data Set, Aggregation with User Preference Data





