`mongoimport`

On this page

MAC OSX SIERRA AND GO 1.6 INCOMPATIBILITY:

Users running on Mac OSX Sierra require the 3.2.10 or newer version of mongoimport.

Synopsis

The `mongoimport` tool imports content from an Extended JSON, CSV, or TSV export created by `mongoexport`, or potentially, another third-party export tool.

See the mongoexport document for more information regarding `mongoexport`, which provides the inverse "exporting" capability.

Run `mongoimport` from the system command line, not the `mongo` shell.

Considerations

WARNING:

Avoid using `mongoimport` and `mongoexport` for full instance production backups. They do not reliably preserve all rich BSON data types, because JSON can only represent a subset of the types supported by BSON. Use `mongodump` and `mongorestore` as described in MongoDB Backup Methods for this kind of functionality.

`mongoexport` and `mongoimport` uses the strict mode representation for certain types.

`mongoimport` supports data files that are UTF-8 encoded. Using other encodings will produce errors.

Required Access

In order to connect to a `mongod` that enforces authorization with the `--auth` option, you must use the `--username` and `--password` options. The connecting user must possess, at a minimum, the `readWrite` role on the database into which they are importing data.

Options

Changed in version 3.0.0: `mongoimport` removed the `--dbpath` as well as related `--directoryperdb` and `--journal` options. To use `mongoimport`, you must run `mongoimport` against a running `mongod` or `mongos` instance as appropriate.

`mongoimport`

`--help`

Returns information on the options and use of `mongoimport`.

`--verbose, -v`

Increases the amount of internal reporting returned on standard output or in log files. Increase the verbosity with the `-v` form by including the option multiple times, (e.g. `-vvvvv`.)

`--quiet`

Runs `mongoimport` in a quiet mode that attempts to limit the amount of output.

This option suppresses:

- output from database commands
- replication activity
- connection accepted events
- connection closed events

`--version`

Returns the `mongoimport` release number.

`--host <hostname><:port>, -h <hostname><:port>`

Default: localhost:27017

Specifies a resolvable hostname for the `mongod` to which to connect. By default, the `mongoimport` attempts to connect to a MongoDB instance running on the localhost on port number `27017`.

To connect to a replica set, specify the `replSetName` and a seed list of set members, as in the following:

```
<replSetName>/<hostname1><:port>,<hostname2><:port>,<...>
```

You can always connect directly to a single MongoDB instance by specifying the host and port number directly.

Changed in version 3.0.0: If you use IPv6 and use the `<address>:<port>` format, you must enclose the portion of an address and port combination in brackets (e.g. `[<address>]`).

`--port <port>`

Default: 27017

Specifies the TCP port on which the MongoDB instance listens for client connections.

`--ipv6`

Removed in version 3.0.

Enables IPv6 support and allows `mongoimport` to connect to the MongoDB instance using an IPv6 network. Prior to MongoDB 3.0, you had to specify `--ipv6` to use IPv6. In MongoDB 3.0 and later, IPv6 is always enabled.

`--ssl`

New in version 2.6.

Enables connection to a `mongod` or `mongos` that has TLS/SSL support enabled.

Changed in version 3.0: Most MongoDB distributions include support for TLS/SSL. See Configure mongod and mongos for TLS/SSL and TLS/SSL Configuration for Clients for more information about TLS/SSL and MongoDB.

Changed in version 3.4: If `--sslCAFile` is not specified when connecting to an TLS/SSL-enabled server, the system-wide CA certificate store will be used.

`--sslCAFile <filename>`

New in version 2.6.

Specifies the `.pem` file that contains the root certificate chain from the Certificate Authority. Specify the file name of the `.pem` file using relative or absolute paths.

Changed in version 3.0: Most MongoDB distributions include support for TLS/SSL. See Configure mongod and mongos for TLS/SSL and TLS/SSL Configuration for Clients for more information about TLS/SSL and MongoDB.

Changed in version 3.4: If `--sslCAFile` is not specified when connecting to an TLS/SSL-enabled server, the system-wide CA certificate store will be used.

> WARNING:
> **Version 3.2 and earlier:** For SSL connections (`--ssl`) to `mongod` and `mongos`, if the `mongoimport` runs without the `--sslCAFile`, `mongoimport` will not attempt to validate the server certificates. This creates a vulnerability to expired `mongod` and `mongos` certificates as well as to foreign processes posing as valid `mongod` or `mongos` instances. Ensure that you *always* specify the CA file to validate the server certificates in cases where intrusion is a possibility.

`--sslPEMKeyFile <filename>`

New in version 2.6.

Specifies the `.pem` file that contains both the TLS/SSL certificate and key. Specify the file name of the `.pem` file using relative or absolute paths.

This option is required when using the `--ssl` option to connect to a `mongod` or `mongos` that has `CAFile` enabled *without* `allowConnectionsWithoutCertificates`.

Changed in version 3.0: Most MongoDB distributions include support for TLS/SSL. See Configure mongod and mongos for TLS/SSL and TLS/SSL Configuration for Clients for more information about TLS/SSL and MongoDB.

Changed in version 3.4: If `--sslCAFile` is not specified when connecting to an TLS/SSL-enabled server, the system-wide CA certificate store will be used.

`--sslPEMKeyPassword <value>`

New in version 2.6.

Specifies the password to de-crypt the certificate-key file (i.e. `--sslPEMKeyFile`). Use the `--sslPEMKeyPassword` option only if the certificate-key file is encrypted. In all cases, the `mongoimport` will redact the password from all logging and reporting output.

If the private key in the PEM file is encrypted and you do not specify the `--sslPEMKeyPassword` option, the `mongoimport` will prompt for a passphrase. See SSL Certificate Passphrase.

Changed in version 3.0: Most MongoDB distributions include support for TLS/SSL. See Configure mongod and mongos for TLS/SSL and TLS/SSL Configuration for Clients for more information about TLS/SSL and MongoDB.

Changed in version 3.4: If `--sslCAFile` is not specified when connecting to an TLS/SSL-enabled server, the system-wide CA certificate store will be used.

`--sslCRLFile <filename>`

New in version 2.6.

Specifies the `.pem` file that contains the Certificate Revocation List. Specify the file name of the `.pem` file using relative or absolute paths.

Changed in version 3.0: Most MongoDB distributions include support for TLS/SSL. See Configure mongod and mongos for TLS/SSL and TLS/SSL Configuration for Clients for more information about TLS/SSL and MongoDB.

Changed in version 3.4: If `--sslCAFile` is not specified when connecting to an TLS/SSL-enabled server, the system-wide CA certificate store will be used.

`--sslAllowInvalidCertificates`

New in version 2.6.

Bypasses the validation checks for server certificates and allows the use of invalid certificates. When using the `allowInvalidCertificates` setting, MongoDB logs as a warning the use of the invalid certificate.

Changed in version 3.0: Most MongoDB distributions include support for TLS/SSL. See Configure mongod and mongos for TLS/SSL and TLS/SSL Configuration for Clients for more information about TLS/SSL and MongoDB.

Changed in version 3.4: If `--sslCAFile` is not specified when connecting to an TLS/SSL-enabled server, the system-wide CA certificate store will be used.

`--sslAllowInvalidHostnames`

New in version 3.0.

Disables the validation of the hostnames in TLS/SSL certificates. Allows `mongoimport` to connect to MongoDB instances even if the hostname in their certificates do not match the specified hostname.

Changed in version 3.0: Most MongoDB distributions include support for TLS/SSL. See Configure mongod and mongos for TLS/SSL and TLS/SSL Configuration for Clients for more information about TLS/SSL and MongoDB.

Changed in version 3.4: If `--sslCAFile` is not specified when connecting to an TLS/SSL-enabled server, the system-wide CA certificate store will be used.

`--sslFIPSMode`

New in version 2.6.

Directs the `mongoimport` to use the FIPS mode of the installed OpenSSL library. Your system must have a FIPS compliant OpenSSL library to use the `--sslFIPSMode` option.

> **NOTE:**
> FIPS-compatible SSL is available only in MongoDB Enterprise. See Configure MongoDB for FIPS for more information.

`--username <username>, -u <username>`

Specifies a username with which to authenticate to a MongoDB database that uses authentication. Use in conjunction with the `--password` and `--authenticationDatabase` options.

`--password <password>, -p <password>`

Specifies a password with which to authenticate to a MongoDB database that uses authentication. Use in conjunction with the `--username` and `--authenticationDatabase` options.

Changed in version 3.0.0: If you do not specify an argument for `--password`, `mongoimport` returns an error.

Changed in version 3.0.2: If you wish `mongoimport` to prompt the user for the password, pass the `--username` option without `--password` or specify an empty string as the `--password` value, as in `--password ""`.

`--authenticationDatabase <dbname>`

Specifies the database in which the user is created. See Authentication Database.

`--authenticationMechanism <name>`

Default: SCRAM-SHA-1

Changed in version 2.6: Added support for the `PLAIN` and `MONGODB-X509` authentication mechanisms.

Changed in version 3.0: Added support for the `SCRAM-SHA-1` authentication mechanism. Changed default mechanism to `SCRAM-SHA-1`.

Specifies the authentication mechanism the `mongoimport` instance uses to authenticate to the `mongod` or `mongos`.

Value	Description
SCRAM-SHA-1	RFC 5802 standard Salted Challenge Response Authentication Mechanism using the SHA1 hash function.
MONGODB-CR	MongoDB challenge/response authentication.
MONGODB-X509	MongoDB TLS/SSL certificate authentication.
GSSAPI (Kerberos)	External authentication using Kerberos. This mechanism is available only in MongoDB Enterprise.
PLAIN (LDAP SASL)	External authentication using LDAP. You can also use `PLAIN` for authenticating in-database users. `PLAIN` transmits passwords in plain text. This mechanism is available only in MongoDB Enterprise.

`--gssapiServiceName`

New in version 2.6.

Specify the name of the service using GSSAPI/Kerberos. Only required if the service does not use the default name of `mongodb`.

This option is available only in MongoDB Enterprise.

`--gssapiHostName`

New in version 2.6.

Specify the hostname of a service using GSSAPI/Kerberos. *Only* required if the hostname of a machine does not match the hostname resolved by DNS.

This option is available only in MongoDB Enterprise.

`--db <database>, -d <database>`

Specifies the name of the database on which to run the `mongoimport`.

`--collection <collection>, -c <collection>`

Specifies the collection to import.

New in version 2.6: If you do not specify `--collection`, `mongoimport` takes the collection name from the input filename. MongoDB omits the extension of the file from the collection name, if the input file has an extension.

`--fields <field1[,field2]>, -f <field1[,field2]>`

Specify a comma separated list of field names when importing csv or tsv files that do not have field names in the first (i.e. header) line of the file.

If you attempt to include `--fields` when importing JSON data, `mongoimport` will return an error. `--fields` is only for csv or tsv imports.

`--fieldFile <filename>`

As an alternative to `--fields`, the `--fieldFile` option allows you to specify a file that holds a list of field names if your csv or tsv file does not include field names in the first line of the file (i.e. header). Place one field per line.

If you attempt to include `--fieldFile` when importing JSON data, `mongoimport` will return an error. `--fieldFile` is only for csv or tsv imports.

`--ignoreBlanks`

Ignores empty fields in csv and tsv exports. If not specified, `mongoimport` creates fields without values in imported documents.

If you attempt to include `--ignoreBlanks` when importing JSON data, `mongoimport` will return an error. `--ignoreBlanks` is only for csv or tsv imports.

`--type <json|csv|tsv>`

Specifies the file type to import. The default format is JSON, but it's possible to import csv and tsv files.

The `csv` parser accepts that data that complies with RFC **RFC 4180**. As a result, backslashes are *not* a valid escape character. If you use double-quotes to enclose fields in the CSV data, you must escape internal double-quote marks by prepending another double-quote.

`--file <filename>`

Specifies the location and name of a file containing the data to import. If you do not specify a file, `mongoimport` reads data from standard input (e.g. “stdin”).

`--drop`

Modifies the import process so that the target instance drops the collection before importing the data from the input.

`--headerline`

If using `--type csv` or `--type tsv`, uses the first line as field names. Otherwise, `mongoimport` will import the first line as a distinct document.

If you attempt to include `--headerline` when importing JSON data, `mongoimport` will return an error. `--headerline` is only for csv or tsv imports.

`--mode insert|upsert|merge`

Default: insert

New in version 3.4.

Specifies how the import process should handle existing documents in the database that match documents in the import file.

By default, `mongoimport` uses the `_id` field to match documents in the collection with documents in the import file. To specify the fields against which to match existing documents for the `upsert` and `merge` modes, use `--upsertFields`.

Value	Description
`insert`	Insert the documents in the import file. `mongoimport` will log an error if you attempt to import a document that contains a duplicate value for a field with a unique index, such as `_id`.
`upsert`	Replace existing documents in the database with matching documents from the import file. `mongoimport` will insert all other documents. Replace Matching Documents during Import describes how to use `--mode upsert`.
`merge`	Merge existing documents that match a document in the import file with the new document. `mongoimport` will insert all other documents. Merge Matching Documents during Import describes how to use `--mode merge`.

`--upsertFields <field1[,field2]>`

Specifies a list of fields for the query portion of the upsert. Use this option if the `_id` fields in the existing documents don’t match the field in the document, but another field or field combination can uniquely identify documents as a basis for performing upsert operations.

Changed in version 3.4: Modifies the import process to update existing objects in the database if they match based on the specified fields, while inserting all other objects. You do not need to use `--mode upsert` with `--upsertFields`.

If you do not specify a field, `--upsertFields` will upsert on the basis of the `_id` field.

To ensure adequate performance, indexes should exist for this field or fields.

`--stopOnError`

Forces `mongoimport` to halt the insert operation at the first error rather than continuing the operation despite errors.

`--jsonArray`

Accepts the import of data expressed with multiple MongoDB documents within a single JSON array. Limited to imports of 16 MB or smaller.

Use `--jsonArray` in conjunction with `mongoexport --jsonArray`.

`--maintainInsertionOrder`

Default: False

If specified, `mongoimport` inserts the documents in the order of their appearance in the input source, otherwise `mongoimport` may perform the insertions in an arbitrary order.

`--numInsertionWorkers int`

Default: 1

New in version 3.0.0.

Specifies the number of insertion workers to run concurrently.

For large imports, increasing the number of insertion workers may increase the speed of the import.

`--writeConcern <document>`

Default: majority

Specifies the write concern for each write operation that `mongoimport` writes to the target database.

Specify the write concern as a document with w options.

`--bypassDocumentValidation`

Enables `mongoimport` to bypass document validation during the operation. This lets you insert documents that do not meet the validation requirements.

New in version 3.2.1.

`--columnsHaveTypes`

New in version 3.4.

Instructs `mongoimport` that the field list specified in `--fields`, `--fieldFile`, or `--headerline` specifies the types of each field.

Field names must be in the form of `<colName>.<type>(<arg>)`. You must backslash-escape the following characters if you wish to include them in an argument: `(`, `)`, and `\`.

type	Supported Arguments	Example Header Field
`auto()`	None.	`misc.auto()`
`binary(<arg>)`	• `base32` (RFC4648 encoding schema) • `base64` (RFC4648 encoding schema) • `hex`	`user thumbnail.binary(base64)`
`boolean()`	None.	`verified.boolean()`
`date(<arg>)`	Alias for `date_go(<arg>)`. Go Language time.Parse format.	`created.date(2006-01-02 15:04:05)`
`date_go(<arg>)`	Go Language time.Parse format	`created.date_go(2006-01-02 15:04:05)`
`date_ms(<arg>)`	Microsoft SQL Server FORMAT format	`created.date_ms(yyyy-MM-dd H:mm:ss)`
`date_oracle(<arg>)`	Oracle Database TO_DATE format.	`created.date_oracle(YYYY-MM-DD HH24:MI:SS)`
`decimal()`	None	`price.decimal()`
`double()`	None.	`revenue.double()`
`int32()`	None.	`followerCount.int32()`
`int64()`	None.	`bigNumber.int64()`
`string()`	None.	`zipcode.string()`

See Import CSV with Specified Field Types for sample usage.

If you attempt to include `--columnsHaveTypes` when importing JSON data, `mongoimport` will return an error. `--columnsHaveTypes` is only for csv or tsv imports.

`--parseGrace <grace>`

Default: stop

New in version 3.4.

Specifies how `mongoimport` handles type coercion failures when importing CSV or TSV files with `--columnsHaveTypes`.

`--parseGrace` has no effect when importing JSON documents.

Value	Description
`autoCast`	Assigns a type based on the value of the field. For example, if a field is defined as a `double` and the value for that field was `"foo"`, `mongoimport` would make that field value a string type.
`skipField`	For the row being imported, `mongoimport` does not include the field whose type does not match the expected type.
`skipRow`	`mongoimport` does not import rows containing a value whose type does not match the expected type.
`stop`	`mongoimport` returns an error that ends the import.

Use

Simple Import

`mongoimport` restores a database from a backup taken with `mongoexport`. Most of the arguments to `mongoexport` also exist for `mongoimport`.

In the following example, `mongoimport` imports the JSON data from the `contacts.json` file into the collection `contacts` in the `users` database.

```
mongoimport --db users --collection contacts --file contacts.json
```

Replace Matching Documents during Import

Changed in version 3.4: In MongoDB 3.4, `--mode upsert` replaces the deprecated `--upsert` option.

With `--mode upsert`, `mongoimport` replaces existing documents in the database that match a document in the import file with the document from the import file. Documents that do not match an existing document in the database are inserted as usual. By default `mongoimport` matches documents based on the `_id` field. Use `--upsertFields` to specify the fields to match against.

Consider the following document in the `people` collection in the `example` database:

```
mongoDB
{
   "_id" : ObjectId("580100f4da893943d393e909"),
   "name" : "Crystal Duncan",
   "region" : "United States",
   "email" : "crystal@example.com"
}
```

The following document exists in a `people-20160927.json` JSON file. The `_id` field of the JSON object matches the `_id` field of the document in the `people` collection.

```
{
   "_id" : ObjectId("580100f4da893943d393e909"),
   "username" : "crystal",
   "likes" : [ "running", "pandas", "software development" ]
}
```

To import the `people-20160927.json` file and replace documents in the database that match the documents in the import file, specify `--mode upsert`, as in the following:

```
mongoimport -c people -d example --mode upsert --file people-20160927.json
```

The document in the `people` collection would then contain only the fields from the imported document, as in the following:

```
{
   "_id" : ObjectId("580100f4da893943d393e909"),
   "username" : "crystal",
   "likes" : [ "running", "pandas", "software development" ]
}
```

Merge Matching Documents during Import

New in version 3.4.

With `--mode merge`, `mongoimport` enables you to merge fields from a new record with an existing document in the database. Documents that do not match an existing document in the database are inserted as usual. By default `mongoimport` matches documents based on the `_id` field. Use `--upsertFields` to specify the fields to match against.

The `people` collection in the `example` database contains the following document:

```
mongoDB
{
   "_id" : ObjectId("580100f4da893943d393e909"),
   "name" : "Crystal Duncan",
   "region" : "United States",
   "email" : "crystal@example.com"
}
```

The following document exists in a `people-20160927.json` JSON file. The `_id` field of the JSON object matches the `_id` field of the document in the `people` collection.

```
{
   "_id" : ObjectId("580100f4da893943d393e909"),
   "username" : "crystal",
   "email": "crystal.duncan@example.com",
   "likes" : [ "running", "pandas", "software development" ]
}
```

To import the `people-20160927.json` file and merge documents from the import file with matching documents in the database, specify `--mode merge`, as in the following:

```
mongoimport -c people -d example --mode merge --file people-20160927.json
```

The import operation combines the fields from the JSON file with the original document in the database, matching the documents based on the `_id` field. During the import process, `mongoimport` adds the new `username` and `likes` fields to the document and updates the `email` field with the value from the imported document, as in the following:

```
{
   "_id" : ObjectId("580100f4da893943d393e909"),
   "name" : "Crystal Duncan",
   "region" : "United States",
   "email" : "crystal.duncan@example.com",
   "username" : "crystal",
   "likes" : [
      "running",
      "pandas",
      "software development"
   ]
}
```

Import `JSON` to Remote Host Running with Authentication

In the following example, `mongoimport` imports data from the file `/opt/backups/mdb1-examplenet.json` into the `contacts` collection within the database `marketing` on a remote MongoDB database with authentication enabled.

`mongoimport` connects to the `mongod` instance running on the host `mongodb1.example.net` over port `37017`. It authenticates with the username `user` and the password `pass`.

```
mongoimport --host mongodb1.example.net --port 37017 --username user --password "pass" --
```

CSV Import

General CSV Import

In the following example, `mongoimport` imports the csv formatted data in the `/opt/backups/contacts.csv` file into the collection `contacts` in the `users` database on the MongoDB instance running on the localhost port numbered `27017`.

Specifying `--headerline` instructs `mongoimport` to determine the name of the fields using the first line in the CSV file.

```
mongoimport --db users --collection contacts --type csv --headerline --file /opt/backups/
```

`mongoimport` uses the input file name, without the extension, as the collection name if `-c` or `--collection` is unspecified. The following example is therefore equivalent:

```
mongoimport --db users --type csv --headerline --file /opt/backups/contacts.csv
```

Import CSV with Specified Field Types

New in version 3.4.

MongoDB 3.4 added support for specifying field types. Specify field names and types in the form `<colName>.<type>(<arg>)` using `--fields`, `--fieldFile`, or `--headerline`.

Consider the following CSV data:

```
Katherine Gray, 1996-02-03, F, 1235, TG9yZW0gaXBzdW0gZG9sb3Igc2l0IGFtZXQsIGNvbnNlY3RldHVy
Albert Gilbert, 1992-04-24, T, 13, Q3VwY2FrZSBpcHN1bSBkb2xvciBzaXQgYW1ldCB0b290c2llIHJvbG
```

The `--fields` option specifies which field type `mongoimport` will use when importing the data into MongoDB.

```
mongoimport --db users --collection contacts --type csv --columnsHaveTypes --fields "name
```

Ignore Blank Fields

Use the `--ignoreBlanks` option to ignore blank fields. For CSV and TSV imports, this option provides the desired functionality in most cases because it avoids inserting fields with null values into your collection.

The following example imports the data from `data.csv`, skipping any blank fields:

```
mongoimport --db users --collection contacts --type csv --file /example/data.csv --ignore
```