

3. On the source instance, run FLUSH TABLES ... FOR EXPORT to quiesce the partitioned table. When a table is quiesced, only read-only transactions are permitted on the table.

mysql> **USE** **test;**

mysql> **FLUSH** **TABLES** **t1** **FOR** **EXPORT;**

FLUSH TABLES ... FOR EXPORT ensures that changes to the named table are flushed to disk so that binary table copy can be made while the instance is running. When FLUSH TABLES ... FOR EXPORT is run, InnoDB generates a .cfg metadata file for each of the table's tablespace files in the schema directory of the table.

mysql> **\!** **ls** ***/path/to/datadir*/test/**

t1#p#p0 .ibd t1#p#p1 .ibd t1#p#p2 .ibd t1#p#p3 .ibd

t1#p#p0.cfg t1#p#p1.cfg t1#p#p2.cfg t1#p#p3.cfg

The .cfg files contain metadata that used for schema verification during the import operation. FLUSH TABLES ... FOR EXPORT can only be run on the table, not on individual table partitions.

4. Copy the .ibd and .cfg files for partition p2 and partition p3 from the source instance schema directory to the destination instance schema directory.

$> **scp** **t1#p#p2.ibd** **t1#p#p2.cfg** **t1#p#p3.ibd** **t1#p#p3.cfg** **destination-server:*/path/to/datadir*/test**

The .ibd and .cfg files must be copied before releasing the shared locks, as described in the next step.

**Note**

If you are importing partitions from an encrypted tablespace, InnoDB generates a .cfp files in addition to a .cfg metadata files. The .cfp files must be copied to the destination instance together with the .cfg files. The .cfp files contain a transfer key and an encrypted tablespace key. On import, InnoDB uses the transfer key to decrypt the tablespace key. For related information, see Section 15.13, “InnoDB Data-at-Rest Encryption” .

5. On the source instance, use UNLOCK TABLES to release the locks acquired by FLUSH TABLES ... FOR EXPORT:

mysql> **USE** **test;**

mysql> **UNLOCK** **TABLES;**

6. On the destination instance, import table partitions p2 and p3:

mysql> **USE** **test;**

mysql> **ALTER** **TABLE** **t1** **IMPORT** **PARTITION** **p2,** **p3** **TABLESPACE;**

 **Note**

 When ALTER TABLE ... IMPORT PARTITION ... TABLESPACE is



**Limitations**

run on subpartitioned tables, both partition and subpartition table names are permitted. When a partition name is specified, subpartitions of that partition are included in the operation.

• The *Transportable* *Tablespaces* feature is only supported for tables that reside in file-per-table tablespaces. It is not supported for the tables that reside in the system tablespace or general tablespaces. Tables in shared tablespaces cannot be quiesced.

• FLUSH TABLES ... FOR EXPORT is not supported on tables with a FULLTEXT index, as full-

text search auxiliary tables cannot be flushed. After importing a table with a FULLTEXT index, run OPTIMIZE TABLE to rebuild the FULLTEXT indexes. Alternatively, drop FULLTEXT indexes before the export operation and recreate the indexes after importing the table on the destination instance.

• Due to a .cfg metadata file limitation, schema mismatches are not reported for partition type or partition definition differences when importing a partitioned table. Column differences are reported.

• Prior to MySQL 8.0.19, index key part sort order information is not stored to the .cfg metadata file used during a tablespace import operation. The index key part sort order is therefore assumed to be ascending, which is the default. As a result, records could be sorted in an unintended order if one table involved in the import operation is defined with a DESC index key part sort order and the other table is not. The workaround is to drop and recreate affected indexes. For information about index key part sort order, see Section 13.1.15, “CREATE INDEX Statement” .

The .cfg file format was updated in MySQL 8.0.19 to include index key part sort order information. The issue described above does not affect import operations between MySQL 8.0.19 server instances or higher.

**Usage** **Notes**

• With the exception of tables that contain instantly added or dropped columns, ALTER TABLE ... IMPORT TABLESPACE does not require a .cfg metadata file to import a table. However, metadata checks are not performed when importing without a .cfg file, and a warning similar to the following is issued:

Message: InnoDB: IO Read error: (2, No such file or directory) Error opening '.\

test\t .cfg', will attempt to import without schema verification

1 row in set (0.00 sec)

Importing a table without a .cfg metadata file should only be considered if no schema mismatches are expected and the table does not contain any instantly added or dropped columns. The ability to import without a .cfg file could be useful in crash recovery scenarios where metadata is not accessible.

Attempting to import a table with columns that were added or dropped using ALGORITHM=INSTANT without using a .cfg file can result in undefined behavior.

• On Windows, InnoDB stores database, tablespace, and table names internally in lowercase. To avoid import problems on case-sensitive operating systems such as Linux and Unix, create all databases, tablespaces, and tables using lowercase names. A convenient way to ensure that names are created in lowercase is to set lower\_case\_table\_names to 1 before initializing the server. (It is prohibited to start the server with a lower\_case\_table\_names setting that is different from the setting used when the server was initialized.)

[mysqld]

lower\_case\_table\_names=1

• When running ALTER TABLE ... DISCARD PARTITION ... TABLESPACE and ALTER TABLE ... IMPORT PARTITION ... TABLESPACE on subpartitioned tables, both partition and subpartition table names are permitted. When a partition name is specified, subpartitions of that partition are included in the operation.

**Internals**

The following information describes internals and messages written to the error log during a table import procedure.

When ALTER TABLE ... DISCARD TABLESPACE is run on the destination instance: • The table is locked in X mode.

• The tablespace is detached from the table.

When FLUSH TABLES ... FOR EXPORT is run on the source instance:

• The table being flushed for export is locked in shared mode.



• The purge coordinator thread is stopped.

• Dirty pages are synchronized to disk.

• Table metadata is written to the binary .cfg file. Expected error log messages for this operation:

[Note] InnoDB: Sync to disk of '"test"."t1"' started.

[Note] InnoDB: Stopping purge

[Note] InnoDB: Writing table metadata to ' ./test/t1 .cfg'

[Note] InnoDB: Table '"test"."t1"' flushed to disk

When UNLOCK TABLES is run on the source instance:

• The binary .cfg file is deleted.

• The shared lock on the table or tables being imported is released and the purge coordinator thread is restarted.

Expected error log messages for this operation:

[Note] InnoDB: Deleting the meta-data file './test/t1.cfg'

[Note] InnoDB: Resuming purge

When ALTER TABLE ... IMPORT TABLESPACE is run on the destination instance, the import algorithm performs the following operations for each tablespace being imported:

• Each tablespace page is checked for corruption.

• The space ID and log sequence numbers (LSNs) on each page are updated.

• Flags are validated and LSN updated for the header page.

• Btree pages are updated.

• The page state is set to dirty so that it is written to disk. Expected error log messages for this operation:

[Note] InnoDB: Importing tablespace for table 'test/t1' that was exported

from host '*host\_name* '

[Note] InnoDB: Phase I - Update all pages

[Note] InnoDB: Sync to disk

[Note] InnoDB: Sync to disk - done!

[Note] InnoDB: Phase III - Flush changes to disk

[Note] InnoDB: Phase IV - Flush complete

**Note**

You may also receive a warning that a tablespace is discarded (if you discarded the tablespace for the destination table) and a message stating that statistics could not be calculated due to a missing .ibd file:

[Warning] InnoDB: Table "test"."t1" tablespace is set as discarded.

7f34d9a37700 InnoDB: cannot calculate statistics for table

"test" ."t1" because the .ibd file is missing . For help, please refer to

<http://dev.mysql.com/doc/refman/8.0/en/innodb-troubleshooting.html>

**15.6.1.4** **Moving** **or** **Copying** **InnoDB** **Tables**

This section describes techniques for moving or copying some or all InnoDB tables to a different server or instance. For example, you might move an entire MySQL instance to a larger, faster server; you might clone an entire MySQL instance to a new replica server; you might copy individual tables to another instance to develop and test an application, or to a data warehouse server to produce reports.



On Windows, InnoDB always stores database and table names internally in lowercase. To move databases in a binary format from Unix to Windows or from Windows to Unix, create all databases and tables using lowercase names. A convenient way to accomplish this is to add the following line to the [mysqld] section of your my.cnf or my.ini file before creating any databases or tables:

[mysqld]

lower\_case\_table\_names=1

**Note**

It is prohibited to start the server with a lower\_case\_table\_names setting that is different from the setting used when the server was initialized.

Techniques for moving or copying InnoDB tables include:

• [Importing Tables](#_bookmark1)

• [MySQL Enterprise Backup](#_bookmark2)

• [Copying Data Files (Cold Backup Method)](#_bookmark3)

• [Restoring from a Logical Backup](#_bookmark4)

**Importing** **Tables**

A table that resides in a file-per-table tablespace can be imported from another MySQL server instance or from a backup using the *Transportable* *Tablespace* feature. See Section 15.6.1.3, “Importing InnoDB

Tables” .

**MySQL** **Enterprise** **Backup**

The MySQL Enterprise Backup product lets you back up a running MySQL database with minimal disruption to operations while producing a consistent snapshot of the database. When MySQL Enterprise Backup is copying tables, reads and writes can continue. In addition, MySQL Enterprise Backup can create compressed backup files, and back up subsets of tables. In conjunction with the MySQL binary log, you can perform point-in-time recovery. MySQL Enterprise Backup is included as part of the MySQL Enterprise subscription.

For more details about MySQL Enterprise Backup, see Section 30.2, “MySQL Enterprise Backup

Overview” .

**Copying** **Data** **Files** **(Cold** **Backup** **Method)**

You can move an InnoDB database simply by copying all the relevant files listed under "Cold Backups" in Section 15.18.1, “InnoDB Backup” .

InnoDB data and log files are binary-compatible on all platforms having the same floating-point number format. If the floating-point formats differ but you have not used FLOAT or DOUBLE data types in your tables, then the procedure is the same: simply copy the relevant files.

When you move or copy file-per-table .ibd files, the database directory name must be the same on the source and destination systems. The table definition stored in the InnoDB shared tablespace includes the database name. The transaction IDs and log sequence numbers stored in the tablespace files also differ between databases.

To move an .ibd file and the associated table from one database to another, use a RENAME TABLE statement:

RENAME TABLE *db1* *.tbl\_name* TO *db2* *.tbl\_name*;

If you have a “clean” backup of an .ibd file, you can restore it to the MySQL installation from which it originated as follows:



1. The table must not have been dropped or truncated since you copied the .ibd file, because doing so changes the table ID stored inside the tablespace.

2. Issue this ALTER TABLE statement to delete the current .ibd file: ALTER TABLE *tbl\_name* DISCARD TABLESPACE;

3. Copy the backup .ibd file to the proper database directory.

4. Issue this ALTER TABLE statement to tell InnoDB to use the new .ibd file for the table: ALTER TABLE *tbl\_name* IMPORT TABLESPACE;

**Note**

The ALTER TABLE ... IMPORT TABLESPACE feature does not enforce foreign key constraints on imported data.

In this context, a “clean” .ibd file backup is one for which the following requirements are satisfied:

• There are no uncommitted modifications by transactions in the .ibd file.

• There are no unmerged insert buffer entries in the .ibd file.

• Purge has removed all delete-marked index records from the .ibd file.

• mysqld has flushed all modified pages of the .ibd file from the buffer pool to the file. You can make a clean backup .ibd file using the following method:

1. Stop all activity from the mysqld server and commit all transactions.

2. Wait until SHOW ENGINE INNODB STATUS shows that there are no active transactions in the database, and the main thread status of InnoDB is Waiting for server activity. Then you can make a copy of the .ibd file.

Another method for making a clean copy of an .ibd file is to use the MySQL Enterprise Backup product:

1. Use MySQL Enterprise Backup to back up the InnoDB installation.

2. Start a second mysqld server on the backup and let it clean up the .ibd files in the backup.

**Restoring** **from** **a** **Logical** **Backup**

You can use a utility such as mysqldump to perform a logical backup, which produces a set of SQL statements that can be executed to reproduce the original database object definitions and table data for transfer to another SQL server. Using this method, it does not matter whether the formats differ or if your tables contain floating-point data.

To improve the performance of this method, disable autocommit when importing data. Perform a commit only after importing an entire table or segment of a table.

**15.6.1.5** **Converting** **Tables** **from** **MyISAM** **to** **InnoDB**

If you have MyISAM tables that you want to convert to InnoDB for better reliability and scalability, review the following guidelines and tips before converting.

**Note**

Partitioned MyISAM tables created in previous versions of MySQL are not compatible with MySQL 8.0. Such tables must be prepared prior to upgrade, either by removing the partitioning, or by converting them to InnoDB. See