

# Index Creation Failures and Errors

Last updated by | Vitor Tomaz | Feb 24, 2023 at 3:27 AM PST

## Contents

- [Issue](#)
- [Error](#)
- [Workaround/Mitigation](#)
- [Other issues related to similar error](#)
- [Monitoring the error](#)
- [Reference articles](#)

Created On	Authored by	Reviewed By
11/02/2022	chakrabortyt	sojaga

## Issue

A backend compatibility issue was encountered recently when the creation of a non-clustered index on a partitioned table of a hyper-scale DB failed with the error 666. The table in question had almost 3.5 Billion records and already had a clustered Index & 3 other non-clustered indexes present. You may receive an error as shown below:

```
Messages
Msg 666, Level 16, State 2, Line 9
The maximum system-generated unique value for a duplicate group was exceeded for index with partition ID 422216236255744. Dropping and re-creating the index may resolve this; otherwise, use another clustering key.
The statement has been terminated.
Completion time: 2022-09-17T00:53:58.0742568+00:00
```

## Error

In addition to the error above- here is the error text Msg 666, Level 16, State 2, Line 25 The maximum system-generated unique value for a duplicate group was exceeded for the index with partition ID. Dropping and re-creating the index may resolve this; otherwise, use another clustering key.;

## Workaround/Mitigation

Customers hitting this problem often are recommended to try running the index creation process at compatibility level 160 (instead of the current compatibility level) as the compatibility level 150 or below might use a spool that is directly associated with uniqueifier identifiers that have a max limit of 2,147,483,648. If this limit is reached the index creation fails with the error mentioned above. (Please note that compatibility level could be just one of the factors that may govern the use of a spool) Here is the difference in explain plans when we use compatibility level 160 vs compatibility level 150 (In the current case), notice the index spool (Highlighted in blue)



For database tables having billions of rows even using compatibility level, 160 may not be sufficient as the index creation process may not encounter the error 666 mentioned above but can eventually time out if the create index transaction exceeds 1 TB in the generated transaction log.

The workaround for the same is to make index creation Online & resumable by specifying `ONLINE=ON`, `RESUMABLE=ON`. With this, the operation will use many smaller transactions, and it will be possible to resume it from the failure point if it fails for any other reason. Using resumable operations is one of the best practices with large tables. It should also be noted that the database scoped configuration `ELEVATE_ONLINE` is set to `OFF` during the index creation process (The default value of `ELEVATE_ONLINE` is `OFF`).

In some cases, if the customer has concerns about changing the compatibility level to 160 for the database, we can also recommend them to change the compatibility level of the DB to 160 just before the index creation process, then trigger the create Index statement and then change the compatibility level of the DB back to 150 (After verifying the Index creation process has started successfully).


## Other issues related to similar error

Please note that resuming a failed index creation is a manual operation. You can do that by re-executing the original `CREATE INDEX` command, it will pick up from the point where it failed. Note that by default, paused resumable operations time out after 24 hours. You can control that using the `PAUSED_RESUMABLE_INDEX_ABORT_DURATION_MINUTES` database-scoped configuration.

It is worthwhile keeping in mind, that for some Big partitioned tables, the rate of progress of the index creation process could be slow if the table has fewer populated partitions. In the test case seen above, the table only had 2 populated partitions & the current plan was running with parallelism (DOP 8), allocating one thread per partition for a total of 8 (plus one coordinator). But there were only two partitions and since one of them is smaller, it had already been processed. So effectively this was running single-threaded now, reading data from the single remaining partition. The index creation process is usually faster if the data is less skewed in partitions in which case the process could even be made faster by adding `MAXDOP=16` to the create index statement.

## Monitoring the error



It is always recommended to monitor such index creation processes periodically to ensure they are progressing well and are not being blocked by any other processes. Here are some of the DMVs that can help monitoring such an index creation process:

1. The actual plan for the CREATE INDEX while it is running can be captured, using [sys.dm\\_exec\\_query\\_statistics\\_xml \(Transact-SQL\) - SQL Server](#) 
2. It is always recommended to Check resource utilization in sys.dm\_db\_resource\_stats a few minutes after starting to create the index. If anything (other than memory and log IO) is above 80%, you may want to increase cores even higher.
3. The progress of the Index creation can be tracked via sys.index\_resumable\_operations. A sample output looks like this:

object_id	index_id	name	sql_text	last_max_dop_used	partition_number	state	state_desc	start_time	last_pause_time	total_execution_time	percent_complete	page_count
1	10		CREATE NONCLUSTERED INDEX	0	NULL	0	RUNNING	2022-10-12 12:45:05.487	NULL	7	3.49195579760483	0

4. More info on waits can be obtained by querying the DMV sys.dm\_exec\_session\_wait\_stats
5. The DMV sys.dm\_exec\_requests indicates if the create Index statement is blocked.
6. If we want to check on any wait types and blocking, the DMV sys.dm\_os\_waiting\_tasks can be very helpful.

## Reference articles

- [Uniqueifier considerations and error 666](#) 
- [Partitioned tables and indexes](#) 

**How good have you found this content?**

