TopicalTraceFlags Documentation

Release 1.0

Aaron Morelli

May 31, 2017

How to Use

Welcome to Topical Trace Flags, a collection of SQL Server trace flags organized by area of the SQL engine. The goal of this project is to present the full set of publicly-known documented and undocumented trace flags in a usable and intuitive way. The content is always a work-in-progress and your contributions are always welcome! Please review the disclaimers and conventions below and the navigational bar to the left to familiarize yourself with the collection's limitations and it's organization. Thanks for dropping by!

Editor/Primary compiler: Aaron Morelli

License: Topical Trace Flags is licensed under the MIT license.

Short Disclaimer: The majority of these flags are undocumented by Microsoft, and therefore unsupported. This repository has neither the authority nor ability to vouch for complete accuracy in the descriptions provided. The contributors to this repository assumes no responsibility for any negative (or positive!) consequences arising from the use or misuse of these trace flags in any production or non-production environment. For more information, see the full *Disclaimers* page. Use at your own risk.

Flags are organized into 3 main parent categories (and an "other" category), with specific pages for each subject area within a category. Each page contains a brief description of that subject area, followed by a table with short descriptions of the flags. This table is followed by more verbose descriptions for each flag, including important links. For a full description of the conventions of this repository, see the *Conventions* page.

To see additions (to this repository, not necessarily to the SQL Server product) in the last 6-12 months, check out the *Recent Additions* page.

CHAPTER 1

Recent Additions

The following additions or modifications are newer to this repository. Checking this page periodically can serve as a "news digest" for new flags in the product, or changes to existing flags. Note that the same Disclaimers apply here as to the repository as a whole: this content is only a best-effort. Flags announcements could slip by unnoticed to the maintainer and contributors of this repo. Doing your own homework is essential for any big decisions you are making to your environments.

CHAPTER 2

Disclaimers

The numerous trace flags in SQL Server, both documented and undocumented, provide a powerful way to modify the behavior of the SQL Server engine and to explore its inner workings. A familiarity with the the more commonly-used trace flags can be helpful in handling various performance edge cases or unwanted out-of-the-box phenomena. Also, it is difficult to go very far in the blogosphere w/o encountering undocumented trace flags. A master SQL Server professional understands trace flags well enough to use them properly and avoid abuse.

However, the power of trace flags and the number of undocumented flags (or even the number of documented but not well-understaood flags) makes it imperative to always tread with caution. Most of the flags in this repository should never be used in a production environment w/o the direct guidance of Microsoft support.

When using this repository, keep the following limitations in mind:

- The descriptions chosen for each flag in this repository are summaries created by the contributors and cannot be guaranteed to completely describe every behavioral impact of the flag being described.
- These summary descriptions usually derive from existing content on the internet (reachable via the links) and only rarely reflect the individual experiences of the contributors to this repository. The accuracy of these descriptions can only be as good as its sources, and no guarantee is ever made that the sources are accurate.
- This repository makes no claim, nor is able to make the claim, that it comprehensively lists every trace flag in the SQL Server product. Many flags may exist only in internal builds inside Microsoft. Additionally, the numeric gaps between flags may or may not indicate the presence of undiscovered flags. There is simply no way for anyone outside of direct employees of Microsoft with source code access to create a truly "comprehensive repository".
- There is no guarantee that what a flag does today will be what it does in tomorrow's build of the product. The fundamental behavior for a given flag rarely changes, but the side effects of flags can change often as new features are introduced and old ones are deprecated.
- No attempt has been made to define the lifespan (earliest and latest builds) for a given trace flag and its effect. The number of flags and the number of SQL Server versions in "recent/modern" history make this effort completely unfeasible. The lifespan of a given flag may sometimes be determined from the nature of the flag, or from various online resources.

CHAPTER 3

Conventions

CHAPTER 4

Using Trace Flags

Most (but not all) trace flags can be enabled at SQL Server startup by using the –T (capital letter) startup option. However, most (but not all) trace flags can also be enabled at startup by using the –t (lowercase letter) startup option. (Besides the upper/lowercase difference, the syntax is otherwise identical). However, BOL indicates that other, unknown trace flags are enabled by the lowercase option, and thus the uppercase method should be used except under the direction of Microsoft support.

Be careful when adding Trace Flags (or other startup options) to the SQL Server via the Configuration Manager GUI.

Microsoft employee Nacho Alonso Portillo notes the following about the DBCC TRACEON command:

- The valid range for trace flags for 2008 R2 and before is -1 to 9299. Note that "9299" as an upper limit is clearly dated, as there are now trace flags with #'s higher than 9299, such as the well-known 9481, and apparently even a 10202 in "vNext" (the version after SQL 2014).
- However, "-1" has the well-known special meaning of enabling/disabling the flag globally.
- The "-1" wildcard can appear in any position in the DBCC TRACEON syntax, though by convention most online use of it always pushes it to the back of the command, after other TF #'s.
- Some flags can only be used with the "-T" startup option, and some can only be used with the DBCC TRACEON command. Attempting to enable "-T"-only flags with DBCC TRACEON will result in an error.

Trace flags that affect query plan or query execution behavior can be enabled for individual queries (rather than session- or system-wide) via the QUERYTRACEON query hint. Not all trace flags are officially supported with QUERYTRACEON; KB2801413 documents those that are.

KB974006 notes: "If DBCC TRACEONTRACEOFF is used this does not regenerate a new cached plan for stored procedures. Plans could be in cache that were created without the trace flag"

TF 4199 (officially described in KB974006) is a special flag:

- Starting with SQL 2000 SP3, the query optimizer team has released all hotfix code under a policy such that the fix remains disabled until TF 4199 is enabled, unless that fix corrects a bug related to incorrect results or assertions or the like.
- SQL Sasquatch (Blog | Twitter) has a short blog series on TF 4199, in which he notes that there are risks inherent in enabling TF 4199 system-wide. This makes Microsoft's official support for the QUERYTRACEON individual query hint even more important and useful.

- In the same blog series, SQL Sasquatch points out that Microsoft has released several fixes recently that seem to violate the TF 4199 policy described above.

A 5-year-old blog post by Andrew Kelly relays some information from Paul Randal about how trace flags are categorized. This provides some level of insight into how Microsoft categorizes (or perhaps categorized at some point in the past) their trace flags, though even a cursory glance through the lists below show that many flags do not fall into any of the below categories:

- 25xx, 52xx are DBCC related
- 8xx are buffer pool
- 36xx are SQL Server general 'run-time'
- 6xx are Storage Engine access methods
- 12xx are lock manager
- 14xx are database mirroring
- 30xx, 31xx, 32xx are backup/restore
- 55xx are FILESTREAM
- 73xx, 74xx are query execution
- 75xx are cursors
- 82xx are replication

CHAPTER 5

Other Important Repositories

Official Trace Flag Documentation from Microsoft:

- Current (At this time Microsoft makes no distinction betweeen a "2014" version of the page or a "2016" version.)
- 2012
- 2008 R2
- 2008
- 2005
- TF 4199
- KB920093 (Tuning options for high-performance workloads)
- KB2801413 (Flags supported for use with QUERYTRACEON)

Print Sources:

- Kalen Delaney's SQL Server 2008 Internals referred to in shorthand as "Kalen2008"
- Ken Henderson's SQL Server 2005 Practical Troubleshooting referred to in shorthand as "Khen2005".

Notable unofficial trace flag repositories:

- Technet Wiki article
- SQL Server Central
- SQL Server Central 2000 and 7.0
- SQLService.se
- Amit Banerjee
- Database-Wiki
- SQL Handle TF4199 series
- Paul Randal presentation

CHAPTER 6

How to Contribute

Topical Trace Flags is hosted on GitHub, and operates under the assumption that anyone in the SQL Server community should be able to add value to the repository in an easy way and receive credit for doing so. The repository is "coded" via the reStructuredText markup language, but you don't have to learn this markup to make changes. Even submitters of pull requests can usually just copy an existing flag's content and then change it, mimicking the various markup tags from the copied text.

To request a change, first search the repo using the Search textbox in the upper-left window. [Annoyingly] you apparently must be on the root page of the repo for search to check all sub-pages, so click the "TopicalTraceFlags" Home button in the upper-left first. You can also use the callout box in the lower left corner of the browser or this search link.

Then, review the search results and confirm which page holds the full definition for the flag (versus the pages that only hold "SeeAlso" references to the flag). Determine the change(s) you want made, ensuring they comply with the *Guidelines for changes* section below.

Once you understand the flag's existing content (or that the flag is completely new), you have several options: 1) non-GitHub methods like email or Twitter, 2) GitHub issues, or 3) GitHub pull requests.

Submitting through email or Twitter

You can contact the current project maintainer (Aaron Morelli) via Twitter or by email: aaron no-intervening-characters morelli at zoho daught com. Please include the following information:

- The flag number
- One or more important links to the flag (see guidelines below). Feel free to send all links you're aware of and the maintainer will decide which are valuable enough to include in the repo.
- (optional) How you would prefer to be identified as a contributor (e.g. your email address, Twitter handle, GitHub, etc). If omitted, we'll just use the method with which you contacted us.

We'll generate a pull request in the repository specifically for your submitted content, and in the comments for that PR identify you as the submitter. Additionally, the reStructuredText source code will contain a reST comment (not visible to the end user) like so:

```
.. Submitted by: (Twitter) @FlagSubmitter
```

Creating a GitHub Issue

If you have a GitHub account, you may find it easier (and more open-source-y) to create either an issue or a pull request. (You could do both if you really want.) To create an issue, once you find the page that should be edited from the search you conducted above (the page that holds the full def for the flag):

1. Click on the "Edit on GitHub" button in the upper right of the ReadTheDocs version of the page. This takes you to GitHub, specifically to the reStructuredText (.rst) file that corresponds to the web page you just left.
2. Click the "Copy path" button near the upper right.
3. Click the "Issues" tab near the upper left, and then click "New issue"
4. Enter a short description (including the flag number) in the title, and in the comment box, paste the filepath you just copied and any links you have for this flag. If you want to specify (or correct) text for the description of the flag, use quotation marks to specify exactly what you want.
5. Submit the issue.

(For more info see GitHub's help on submitting an issue).

We'll generate a pull request in the repository specifically for your submitted content, and in the comments for that PR identify you as the submitter. We'll also link the PR to that Issue number, and the reStructuredText source code will contain a reST comment (not visible in the web page) as mentioned above under *Submitting through email or Twitter*.

Creating a GitHub pull request

If you really want control over the change, have a large number of changes to submit, or just want your GitHub user name in the commit history of the project, a GitHub pull request is the way to go. Assuming you're logged in to GitHub, once you find the page that should be edited from the search you conducted above (the page holding the full def for the flag, see above):

1. Click on the "Edit on GitHub" button in the upper right of the page. This takes you to GitHub, specifically to the reStructuredText (.rst) file that corresponds to the web page you just left. Make a note of which file you need to edit.
2. Click on the "Fork" button in the upper right.
3. In your forked repository, in the **master** branch, make the changes to the .rst file you noted above.
4. Submit a pull request.

Feel free to add a reST comment (only visible in the source file) just above the flag you're adding/modifying with identifying info, like so:

```
.. Submitted by: (Twitter) @FlagSubmitter
```

We'll review your PR, discuss back-and-forth if any clarifications are needed, and then merge in to the master branch on our side.

Note: This is a simple project, so you'll almost always make your changes to the master branch in your fork. A more complicated series of changes or re-working of the structure may warrant the creation of a new branch.

Communicate with us (preferably via an issue) if you think this situation applies.

Warning: Never make your changes or submit your pull request for a branch beginning with "DONOTUSE". The current maintainer uses these as private feature branches and pushes to them from his laptop regularly to save WIP, then rebases later to clean up history before merging to master.

Guidelines for changes

1. Normally, every flag should have one or more links to publicly-available content.

1. If you are a Microsoft employee with source code access or a well-known figure in the SQL Server community with a reputation for accurate, detailed internals information (e.g. Paul White, Paul Randal, Dima Piliugin, etc) and you want to provide never-before-seen info on new or existing flags, we welcome that even if there's no public link.

2. Citations from books (e.g. the SQL Server Internals series) are fine as well, as long as an exact quote and a page number is provided.

3. If you *aren't* a "1.A" type of person, but have a blog post with examples of new flags or new behavior for existing flags, the flag w/that link is certainly welcome! We just may caveat the flag in some way to highlight the uncertainty around the its functionality.

2. For existing flags, new content should add something not already obvious from the existing content. (This repository is not attempting to document every blog post that uses Trace Flag 1117!) If a link has an unusual usage of the flag, an interesting side-effect, or some other novel addition to the flag's current content in the repository, then it will certainly be welcome.

3. A link to a KB article or a blog by someone at Microsoft, or a "super-guru", is more valuable than a forum conversation or a post by someone at the fringes of the SQL Server community. Of course, if their post adds value in some other way (creative use, interesting side effects, etc) then it should be included along with any KB article or Microsoft blog post. And if the only link for a flag is to a forum post or an unknown blogger, the repository is richer for it. (There just may need to be a caveat about the uncertainy.)

Any questions? Contact us through Twitter, or send us an email! (aaron no-intervening-characters morelli at zoho daught com). We look forward to your content!

CHAPTER 7

Change Log

Going forward (now that the project is on GitHub), this log only holds significant, broader changes to the repository. Individual flag additions and modifications will not be logged here. Rely on the git commit history (and in some cases, tagging) for more granular analysis.

Dates are in U.S. format (mm/dd).

2016.11.14 Begin migration to GitHub/reStructuredText.

2016.09.22 Reviewed KB articles up to and including 2012 SP2 CU14, 2012 SP3 CU5, 2014 SP1 CU8, and 2016 RTM CU2

2016.04.27 Reviewed KB articles up to and including SQL 2012 SP2 CU11, SQL 2012 SP3 CU2, SQL 2014 RTM CU13, and SQL 2014 SP1 CU6.

2015.06.23 Reviewed KB articles up to and including 2014 RTM CU8 and 2014 SP1 CU1. Cleaned up a few sections and adjusted section spacing for pagination aesthetics.

2015.01.21 Reviewed KB articles (for trace flags) up to and including: SQL 2014 RTM CU5, SQL 2012 SP1 CU14, and SQL 2012 SP2 CU4.

2014.11.12 Reviewed KB articles (for trace flags) up to and including: SQL 2005 SP4 CU3 and SQL Server 2008 R2 SP3.

2014.10.21 Reviewed KB articles (for trace flags) up to and including: SQL 2014 RTM CU4, SQL 2012 RTM CU11, SQL 2012 SP1 CU 12, and SQL 2012 SP CU2.

2014.04.27 Category reorganization/refinement; filled out descriptions for a large # of flags.

2014.03.10 Incorporated TFs resulting from a search of the Microsoft Knowledge Base (KB).

2013.12.20 Incorporated flag and link info from both Banerjee pages, the Technet Wiki article, the Database-Wiki article, and the SQLService.se article into this document.

2013.12.19 Incorporated TFs from all Paul White blog posts into this doc.

2013.10.25 Re-categorized most flags and shortened many descriptions; added links, modified intro.

2012.06.08 Significant formatting changes (font highlighting, restructured several tables, added TOC).

2012.04.09 Document creation.

CHAPTER 8

Statistics and Cardinality Estimation

Flags in this category fall into the below groupings:

1. *Cardinality Estimation Behavior* –> Flags that affect the way that cardinality estimation is done by the query optimizer
2. *Informational, Stats Update, and Other* –> Return info about the cardinality estimation process during optimization, or manipulate or provide information about statistics objects
3. *Fixes and Past Relevance* –> Fix flags and flags which are no longer relevant.

TODO: add a SeeAlso in the security section to here for 9485

Short Descriptions

Flag	Short Description
See Also	
205	Reports to error log when recompile occurs due to auto-update of stats.
8612	Adds cardinality info to query opt trees that are output by by 8605/8606/8607.
8666	Returns hidden nodes in query plan XML including stats usage and thresholds.
9348	Applies a row limit (based on card est) to whether bulk insert is attempted or not.
...	
CE Behav	
2301	Old CE: use a different set of mathematical assumptions during estimation.
* *2312*	Forces the use of the new cardinality estimator during optimization.
2324	Disables the creation of “implied predicates”.
2328	Reverts const/const comparison (incl. parms/vars) estimations to SQL2K guesses.
* *2389*	Enables optimizer tracking of ascending keys and subsequent recompilation behavior.

Continued on next page

Table 8.1 – continued from previous page

Flag	Short Description
* *2390*	Similar to 2390 and the ascending key problem. See description below for more.
2453	Applies recompile thresholds for temp tables to table variables
4137	Adjusts calculation used for multiple AND predicates; effective w/correlation.
4138	Disables rowgoal logic for queries that contain TOP, OPTION(FAST N), IN, or EXISTS.
* *4139*	Enables "quick stats" histogram amendments even if key col isn't branded ascending.
9471	New CE: Use min selectivity for both conjunctive (AND) and disjunctive (OR) preds.
9472	New CE: Assume independence for multiple WHERE predicates.
9476	New CE: Use simple containment (old CE default) instead of base containment.
9479	New CE: Force use of "simple join" estimation algorithm
* *9481*	New CE: Force use of the old cardinality estimation model.
9482	New CE: Turn off "overpopulated primary key" CE adjustment
9483	New CE: Forces QO to create non-persisted filtered stat objects.
9488	New CE: Reverts the estimation for multi statement TVFs back to 1 row.
9489	New CE: turn off new CE ascending key logic.
...	
Other	
2309	Adjusts DBCC SHOW_STATISTICS output, showing info for partial statistics.
* *2363*	New CE: Outputs info about stats used and resulting cardinality estimates.
* *2371*	Lowers relative threshold for stats-based recompilation as table gets larger.
2388	Adjusts DBCC SHOW_STATISTICS output to show (not-)ascending status of stats obj.
7471	Allows multiple statistics objects on the same table to be updated concurrently.
8721	Prints info to error log when AutoStat occurs.
9204	Prints which stats objects the QO found interesting and then fully loaded.
9292	Prints which stats objects the QO found interesting and loaded the header.
9485	Disables SELECT permission for DBCC SHOW_STATISTICS.
...	
Fix/PastRel	

Cardinality Estimation Behavior

2301 `Doc2014` BOL 2014: "Enable advanced decision support optimizations". 920093 adds: "Makes your optimizer work harder by enabling advanced optimizations that are specific to decision support queries, applies to processing of large data sets." IJose_1 lists 5 fundamental mathematical assumptions the optimizer makes when using this flag, and contrasts these assumptions with behavior without this flag.

- Integer Modelling for values falling between histogram steps. Helps with inequality filters.
- Comprehensive Histogram Usage even when the cardinality estimate for a relation drops below the number of steps in the histogram.
- Base Containment Assumption (see blog post)
- Comprehensive Density Remapping changes how estimation is done when a CONVERT function call is present
- Comprehensive Density Matching allows equi-joined columns to be considered equivalent for certain estimation operations.

Database-Wiki adds that the flag "discourages order-preserving parallelism", especially parallel merge join. It also claims that the density remapping logic is also relevant to UPPER and LOWER (and CAST, of course)

function calls.

> **Warning:** IJose adds: "...compile times will increase, and in some cases memory consumption can increase dramatically."

920093 | IJose_1 | Dima_1 | Connect_1

2312 `Doc2014` BOL 2014: "Enables you to set the query optimizer cardinality estimation model to the SQL Server 2014 through SQL Server 2016 versions, dependent of the compatibility level of the database." AKA force use of the new CE.

New CE Whitepaper | 2801413 | Nevarez_1

2324 Disables the creation of "implied predicates". Implied predicates can be safely, mathematically inferred by other criteria in the query and added to the internal representation of the query to assist in cardinality estimation and various other optimizer transforms.

SQLPerf_1

2328 Reverts back to SQL 2000 behavior when comparing 2 constants ("constants" here includes parameters or variables whose values are known at compile time). The SQL 2000 behavior uses a guess formula, whereas the SQL 2005+ behavior uses the compile-time values and the statistics object(s) to generate a cardinality estimate. Dima demonstrates this selectivity guess (a call to CScaop_Comp::GuessSelect)

IJose_2 | Dima_2

2389 `Doc2014` BOL 2014: "Enable automatically generated quick statistics for ascending keys (histogram amendment). If trace flag 2389 is set, and a leading statistics column is marked as ascending, then the histogram used to estimate cardinality will be adjusted at query compile time." Itzik has very insightful comments about this flag and the ascending key problem under both the old and new CE. Kejser notes that (at the time of his posts) this TF doesn't work with partitioned tables, and has his own solution. See also *9489* (Dima). *(See also 2388, 2390*, and *4139)*

Hotfixes: 922063, 929278, 2952101 | Itzik_1 | IJose_3 | Stellato_2 | Kejser Part 1 | Kejser Part 2 | Nevarez_2

2390 `Doc2014` BOL 2014: "Enable automatically generated quick statistics for ascending or unknown keys (histogram amendment). If trace flag 2390 is set, and a leading statistics column is marked as ascending or unknown, then the histogram used to estimate cardinality will be adjusted at query compile time." Closely tied to *2389*, *4139*, and the ascending key problem.

> **Warning:** Read Ian Jose's blog entry carefully before using 2390.

IJose_3

2453 `Doc2014` BOL 2014: "Allows a table variable to trigger recompile when enough number of rows are changed." Applies familiar temp table rowcount recompilation thresholds to table variables. Bertrand points out important caveats.

2952444 | 2012SP2 | CSS_1 | Bertrand

4136 `Doc2014` BOL 2014: "Disables parameter sniffing unless OPTION(RECOMPILE), WITH RECOMPILE or OPTIMIZE FOR value is used." There are now a number of ways for disabling parameter sniffing:

- This flag
- The USE HINT option **DISABLE_PARAMETER_SNIFFING**
- The database-scoped option **PARAMETER_SNIFFING**

- The OPTIMIZE FOR UNKNOWN query hint

Dima notes that 4136 has no effect on parameter sniffing's "cousin", runtime constant sniffing.

980653 | USE HINT | DB SCOPED CONFIG | Dima_10

4137 `Doc2014` BOL 2014: "Causes SQL Server to generate a plan using minimum selectivity when estimating AND predicates for filters to account for correlation, under the query optimizer cardinality estimation model of SQL Server 2012 and earlier versions. Beginning with SQL Server 2016 SP1, to accomplish this at the query level, add the **ASSUME_MIN_SELECTIVITY_FOR_FILTER_ESTIMATES**. USE HINT" The BOL USE HINT link notes that this hint is *equivalent* to 4137 (old CE) and *similar* to *9471* (new CE).

Paul White covers "Minimum Selectivity" well, and also points out that 4137 only applies min selectivity to conjunctive (AND) predicates, while *9471* applies it to both conjunctive and disjunctive (OR) predicates.

2658214 | USE HINT | PWhite_1 | Dima_3 | Connect_2

4138 `Doc2014` BOL 2014: "Causes SQL Server to generate a plan that does not use row goal adjustments with queries that contain TOP, OPTION (FAST N), IN, or EXISTS keywords. Beginning with SQL Server 2016 SP1, to accomplish this at the query level, add the **DISABLE_OPTIMIZER_ROWGOAL** USE HINT."

The 2 interesting StackExch links show Martin Smith at work.

2667211 | USE HINT | PWhite_2 | PWhite_3 | StackExch_1 | StackExch_2

4139 `Doc2014` BOL 2014: "Enable automatically generated quick statistics (histogram amendment) regardless of key column status. If trace flag 4139 is set, regardless of the leading statistics column status (ascending, descending, or stationary), the histogram used to estimate cardinality will be adjusted at query compile time. Beginning with SQL Server 2016 SP1, to accomplish this at the query level, add the **ENABLE_HIST_AMENDMENT_FOR_ASC_KEYS** USE HINT."

The KB introduces the flag as a fix flag for a case where 90% of newly-inserted values were NOT higher than the highest key value. However, the BOL description indicates an intent for the flag to be used in any situation where out-of-bounds values are too costly.

> **Warning:** 3192117 notes this flag can cause access violations on certain SQL 2016 builds.

2952101 | 3192117 | USE HINT

9471 `Doc2014` BOL 2014: "Causes SQL Server to generate a plan using minimum selectivity for single-table filters, under the query optimizer cardinality estimation model of SQL Server 2014 through SQL Server 2016 versions. Beginning with SQL Server 2016 SP1, to accomplish this at the query level, add the **ASSUME_MIN_SELECTIVITY_FOR_FILTER_ESTIMATES** USE HINT." The BOL USE HINT link notes that this hint is *equivalent* to *4137* (old CE) and *similar* to 9471 (new CE).

Paul White covers "Minimum Selectivity" well, and also points out that 4137 only applies min selectivity to conjunctive (AND) predicates, while 9471 applies it to both conjunctive and disjunctive (OR) predicates.

USE HINT | PWhite_1 | Milo_1 | Connect_2

9472 Assumes independence for multiple WHERE predicates in the SQL 2014 CE. Predicate independence was the default before SQL 2014, and thus this flag can be used to emulate pre-SQL 2014 CE behavior in a more specific fashion than TF 9481.

PWhite_1 | Connect_2

9476 `Doc2014` BOL 2014: "Causes SQL Server to generate a plan using the Simple Containment assumption instead of the default Base Containment assumption, under the query optimizer cardinality estimation model of SQL Server 2014 through SQL Server 2016 versions. Beginning with SQL Server 2016 SP1, to accomplish this at the query level, add the USE HINT **ASSUME_JOIN_PREDICATE_DEPENDS_ON_FILTERS**."

Simple containment is the old CE default; base containment is the new CE default. Paul White email: "Ignores the histograms (avoiding coarse alignment) and simply assumes containment at the join."

USE HINT | 3189675 | Dima_1

9479 Dima: "Forces the optimizer to use Simple Join [estimation] even if a histogram is available...will force optimizer to use a simple join estimation algorithm, it may be CSelCalcSimpleJoinWithDistinctCounts, CSelCalcSimpleJoin or CSelCalcSimpleJoinWithUpperBound, depending on the compatibility level and predicate comparison type."

Paul White email: "uses simple containment instead of base containment for simple joins"

Dima_4

9481 `Doc2014` BOL 2014: "Enables you to set the query optimizer cardinality estimation model to the SQL Server 2012 and earlier versions, irrespective of the compatibility level of the database. Beginning with SQL Server 2016 SP1, to accomplish this at the query level, add the USE HINT **FORCE_LEGACY_CARDINALITY_ESTIMATION**."

Note also the **LEGACY_CARDINALITY_ESTIMATION** database-scoped setting.

USE HINT | DB SCOPED CONFIG | 2801413

9482 Turns off the "overpopulated primary key" adjustment that the optimizer might use when determining that a "dimension" table (the schema could be OLTP as well) has many more distinct values than the "fact" table. (The seminal example is where a Date dimension is populated out into the future, but the fact table only has rows up to the current date). Since join cardinality estimation occurs based on the contents of the histograms of the joined columns, an "overpopulated primary key" can result in higher selectivity estimates, causing rowcount estimates to be too low.

Dima_5

9483 Forces the optimizer to create (if possible) a filtered statistics object based on a predicate in the query. This filtered stat object is not persisted. In Dima's example, the filtered stat object is actually created on the join column...i.e. "CREATE STATISTICS [filtered stat obj] ON [table] (Join column) WHERE (predicate column = 'literal')"

Dima_6

9488 Reverts the estimation behavior for multi-statement TVFs back to 1 row instead of the 100-row estimate behavior that was adopted in SQL 2014.

Dima_7

9489 Turns off the new CE logic that handles ascending keys.

Dima_8

Informational, Stats Update, and Other

2309 (Info) In SQL 2014, enables output from a 3rd parameter for DBCC SHOW_STATISTICS such that the partial statistics histogram (for just one partition) is shown.

Stellato_1 | DBIServices_1

2363 (Info) Outputs information regarding statistics information used and derived during the (new CE) optimization process. Dima shows it giving insight into the new "calculator" framework in 2014 for deriving statistics at

intermediate nodes of the tree. It also shows which histograms (statistics objects) were loaded, and acts as a replacement for *9204* and *9292*.

Dima_9 | PWhite_1

2371 `Doc2014` BOL 2014: "Changes the fixed auto update statistics threshold to dynamic auto update statistics threshold. **Note:** Beginning with SQL Server 2016 this behavior is controlled by the engine and trace flag 2371 has no effect." Switches the threshold for a performance-based recompile (for a given stats object) from the default of 20%+500 rows to SQRT(1000*<num rows in table>).

2754171 | SRGolla | SAPonSQLServer

2388 (Info) Changes the output of DBCC SHOW_STATISTICS. Instead of the normal Header/Vector/Histogram output, instead we get a single row that gives information related to whether the lead column of the stat object is considered to be ascending or not. This TF is primarily helpful in watching the state of a stat object change from "Unknown", to "Ascending" (and potentially to "Stationary").

Nevarez_2 | SQLSasquatch_2 | Stellato_2

7471 Allows multiple statistics objects on the same table to be updated concurrently.

3156157 | TigerTeam_1

8721 `Doc2014` **(Info)** BOL 2014: "Reports to the error log when auto-update statistics executes." The KB also describes locks taken by autostats.

195565 | SQLSasquatch_1

9204 (Info) PWhite: for the old CE, gives "the interesting statistics which end up being fully loaded and used to produce cardinality and distribution estimates for some plan alternative or other."

PWhite_4

9292 (Info) PWhite: for the old CE, gives "a report of statistics objects which are considered 'interesting' by the query optimizer when compiling, or recompiling the query in question. For [these] potentially useful statistics, just the header is loaded."

PWhite_4

9485 `Doc2012` BOL 2014: "Disables SELECT permission for DBCC SHOW_STATISTICS."

Fixes and Past Relevance

These flags either are old and irrelevant for modern builds, appear only in CTPs, or enable a fix in a CU but are baselined in a later service pack or release.

CHAPTER 9

Compilation Informational

Flags in this category fall into the below groupings:

1. *Trees and Memos* –> Flag prints one or more of the "trees" constructed during the query optimization process, or the state of the memo at a given point in the optimization process.
2. *Phases and Rules* –> The phases entered by the optimizer and the rules it uses in those phases.
3. *Miscellaneous* –> Resource usage, operator-specific info, and other "miscellaneous" info.
4. *Fixes and Past Relevance* –> Fix flags and flags which are no longer relevant.

All flags in this category are informational (the redundant "Info" tag has been retained for the sake of screenshots and copy-pasting), as far as we know.

Short Descriptions

Flag	Short Description
See Also	
* *2363*	New CE: Outputs info about stats used and resulting cardinality estimates.
8721	Prints info to error log when AutoStat occurs.
9204	Prints which stats objects the QO found interesting and then fully loaded.
9292	Prints which stats objects the QO found interesting and loaded the header.
...	
Trees/Memos	
7352	Final query tree after the post-optimization rewrite phase.
8605	Initial logical tree (input into actual optimization). AKA "converted tree".
8606	Multiple trees: Input, Simplified, Join-collapsed, and before/after Proj Norm

Continued on next page

Table 9.1 – continued from previous page

Flag	Short Description
8607	Optimization output tree, before post-optimization rewrite phase.
8608	Initial Memo structure
8612	Adds cardinality info to trees produced by 8605, 8606, 8607.
8615	Final Memo structure.
8620	Adds Memo arguments to 8619 (which outputs the transformation rules applied)
8621	Adds trees to the transformation rule printout from 8619.
...	
Phases/Rules	
2372	Prints memory utilization for each entered optimization phase.
2373	Prints memory utilization before/after each applied optimizer rule.
8609	Prints task and operation type counts
8619	Prints transformation rules applied during optimization.
8628	Places transformation rules used into the query plan. (8666 exposes in XML)
8675	Prints the query optimization phases entered.
8739	Dima: "Group Optimization Info"
...	
Misc	
205	Reports to error log when recompile occurs due to auto-update of stats.
445	Full func unknown; prints sql text from compilations to error log.
2315	Aaron discovered: appears to output mem allocations during compilation
2318	Aaron discovered: maybe info about join reordering?
2336	Aaron discovered: maybe connects cached page likelihoods, mem status, and costing?
2398	Aaron discovered: outputs info about "Smart Seek costing"
7357	Various info about hash join and match operators.
8666	Causes useful info present in the binary query plan to be exposed in the XML.
8719	(Maybe past relevance) Apparently outputs IO prefetch for NL joins/bookmarks.
...	
Fix/PastRel	

Trees and Memos

7352 (Info) The final query tree after Post-optimization re-write.

PWhite_4 | PWhite_5 | Dima_5 | Dima_6

8605 (Info) The initial logical tree (the input into query optimization). (Paul also calls this the "converted tree" in Part 4 of his series)

PWhite_1 | Nevarez_1

8606 (Info) Displays additional logical trees, including the Input Tree, the Simplified Tree, the Join-collapsed Tree, the "Tree before Project Normalization", and the "Tree after Project Normalization"

PWhite_2 | Dima_1 | Nevarez_1

8607 (Info) Displays the optimization output tree, before Post-optimization rewrite. Has a "Query marked as cachable" note if the plan can be cached.

PWhite_3 | PWhite_6 | PWhite_7 | PWhite_8 | PWhite_9 | Dima_3 | Dima_6 | Nevarez_1

8608 (Info) Shows the initial Memo structure

PWhite_3 | Dima_3 | Dima_4 | Nevarez_2

8612 (Info) Dima: adds cardinality info to the various trees produced by flags 8605, 8606, and 8607.

PWhite_8 | Dima_3 | Dima_7 | Dima_8

8615 (Info) Show the final Memo structure

PWhite_3 | Dima_3 | Dima_4 | Dima_9

8620 (Info) PWhite: "Add memo arguments to 8619"

PWhite_4

8621 (Info) PWhite: "Rule with resulting tree" (use with 8619)

PWhite_4

Phases and Rules

2372 (Info) Nevarez: "shows memory utilization during the different optimization stages."

Nevarez_1 | Dima_8

2373 (Info) Shows memory utilization before and after various optimizer rules are applied (e.g. IJtoIJsel). Provides a way to "trace" what rules are used when optimizing a query.

PWhite_4 | Dima_2 | Dima_8 | Dima_9

8609 (Info) PWhite: "Task and operation type counts"

PWhite_4 | Dima_10

8619 (Info) PWhite: "Apply rule with description"; Dima: "Show Applied Transformation Rules"

PWhite_4 | PWhite_10 | Dima_8

8628 (Info) When used with 8666 (see below), causes extra information about the transformation rules applied to be put into the XML showplan.

Dima_11 | PWhite_11

8675 (Info) Display query optimization phases, along with info (timing, costs, etc) about each phase.

PWhite_3 | PWhite_12 | PWhite_13 | PWhite_9

8739 (Info) Dima: "Group Optimization Info"

Dima_10

Miscellaneous

205 `Doc2014` (Info) BOL 2014: "Reports to the error log when a statistics-dependent stored procedure is being recompiled as a result of auto-update statistics." Its appearance in Randal-SQL-SDB407 indicates that this flag is fairly old.

195565_

445 (Info) Full functionality unknown. Prints "Compile issued:" and then the text of the sql statement being compiled to the SQL error log. Personally confirmed that this still works in SQL 2014 even though it appears to be a very old trace flag. Discovered via SQLService.se

2315 (Info) Aaron: personally discovered. Seems to output memory allocations taken during the compilation process (and maybe the plan as well? "PROCHDR"), as well as memory broker states & values at the beginning and end of compilation.

2318 (Info) Aaron: personally discovered. I've only seen one type of output so far: "Optimization Stage: HEURISTICJOINREORDER". Maybe useful in combo with other compilation trace flags to see the timing of join reordering?

2336 (Info) Aaron: personally discovered. Appears to tie memory info and cached page likelihoods with costing.

2398 (Info) Aaron: personally discovered. Outputs info about "Smart Seek costing": e.g.: "Smart seek costing (75.2) :: 1.34078e+154 , 1".

7357 (Info) Info re: hash operators, including role reversal, recursion levels, Unique Hash optimization usage, hash-related bitmap, etc. For parallel query plans, 7357 does NOT send output to the console window. However, output to the SQL Server error log can be enabled by enabling 3605.

Dima_12 | PWhite_4

8666 (Info) Causes some useful info (including stat object thresholds) already present in the internal representation of a plan to be included in the XML plan output.

Fabiano_1 | DBally_1 | PWhite_14 | PWhite_15

8719 (Info) In SQL 2000, apparently would show IO prefetch on loop joins and bookmarks. I (Aaron) was unable to replicate the query plan behavior on SQL 2012 using the same test, so this flag may be obsolete. (Would be really nice if it wasn't!)

Hanlincrest_

Fixes and Past Relevance

These flags either are old and irrelevant for modern builds, appear only in CTPs, or enable a fix in a CU but are baselined in a later service pack or release.

CHAPTER 10

Compilation Behavioral

All flags in this category modify the compilation process in such a way as to produce alternative query plans. Note that many flags in the *Statistics and Estimation* category also modify compilation behavior; those flags all do so by modifying the cardinality estimation process, whereas all of these flags change non-estimation behaviors.

The information returned falls into the below categories:

1. *Disk IO Optimization* –> Flags whose goal is to improve Disk IO performance (often by promoting sequential IO)
2. *Hashing and Batching* –> Flags that modify hashing operations and flags that adjust batch mode processing
3. *Phases and Timeouts* –> Flags that alter the phases the optimizer enters or its timeout values for short-circuiting phase entry.
4. *Miscellaneous* –> Other, miscellaneous flags.
5. *Fixes and Past Relevance* –> Fix flags and flags which are no longer relevant.

Note that the line between these flags and those in the *Query Execution* category can be a bit grey. Flags in this category affect behaviors that persist in the cached plan between executions, while flags in the *Query Execution* category affect behavior that is normally decided at runtime.

Short Descriptions

Flag	Short Description
See Also	
2301	Modifies estimation mathematical assumptions; may affect non-estimation behavior.
* *2312*	Forces the use of the new CE during optimization, inevitably changing plan shapes.
...	

Continued on next page

Table 10.1 – continued from previous page

Flag	Short Description
Disk Opts	
2332	Forces a sort (for sequential IO) in Insert/Update/Delete plans
2340	Disables nested loops "implicit batch sort" (in post-opt rewrite phase)
8633	Enable prefetch; useful for IO perf in Insert/Update/Delete plans.
8738	(Apparently) disables sorting before a key lookup operator.
8744	Disables pre-fetching for the Nested Loops operator.
8795	Disables sorting for sequential IO in Insert/Update/Delete plans.
9115	Disables both implicit batch sort (TF 2340) and NL prefetching (TF 8744)
...	
Hash/Batch	
2441	Enables the use of hash joins to column store indexes in certain cases.
7359	Disables the internal bitmap used for hash matches and joins.
7497	(Full purpose unknown) Can be used w/7498 to disable "optimized bitmaps"
7498	(Full purpose unknown) Can be used w/7497 to disable "optimized bitmaps"
9347	(Purpose unknown) Appears related to batch mode sorting.
9349	Disables batch mode top sort operator.
9358	Disables batch-mode sort operations.
9453	Disables batch mode, forcing row mode.
...	
Phases/TOs	
8671	Disables the optimizer short-circuit due to "Good Enough Plan Found"
8677	Skips the "Search 1" optimization phase (if applicable)
8757	Skip the Trivial Plan optimization phase
8780	Sets the optimizer timeout value to a very high constant. **Do Not Use!**
8788	Appears to have a similar effect as 8780. **Do Not Use!**
...	
Misc	
2329	Disables the "Few outer rows" dimension table optimization
2335	Causes optimizer to create plans that are more conservative w/memory.
7470	Adjusts calculation for memory requirements for sorts.
8602	Tells optimizer to ignore index hints.
* *8649*	Strongly encourages optimizer to generate a parallel plan.
8690	Prevents the optimizer use of "performance spools"
8692	Forces optimizer to use eager spools when it needs Halloween Protection.
8722	Disables all "other" (besides index and join) query hints.
8746	Among other effects, disables "rowset sharing" optimization.
8755	Disables all join hints.
8758	Among other effects, disables plan rewrites to a single operator plan.
8790	Forces a wide-update plan for any data-changing query.
9130	Disables the pushing of non-sargable filter predicates into seeks or scans.
9348	Applies a row limit to whether bulk insert is attempted or not.
...	
Fix/PastRel	
8687	(Perhaps) disables query parallelism.
8720	In SQL 2000, apparently had the same effect as OPTION(KEEPFIXED PLAN)
9059	Allows optimizer to choose an index seek for numerics of varying precisions.

Disk IO Optimization

2332 PWhite: "Force DML Request Sort (CUpdUtil::FDemandRowsSortedForPerformance)" Appears to be the counterpoint of *8795*.

PWhite_1

2340 `Doc2014` BOL 2014: "Causes SQL Server not to use a sort operation (batch sort) for optimized nested loop joins when generating a plan. Beginning with SQL Server 2016 SP1, to accomplish this at the query level, add the *DISABLE_OPTIMIZED_NESTED_LOOP* USE HINT query hint instead of using this trace flag."

Dima: "Disable Nested Loops Implicit Batch Sort on the Post Optimization Rewrite Phase." Dima's article is useful to contrast NL batch sorting and 2340 from NL prefetching and 8744, and discusses 9115.

2009160 | CSS_1 | Dima_1

8633 PWhite: "Enable prefetch (CUpdUtil::FPrefetchAllowedForDML and CPhyOp_StreamUpdate::FDoNotPrefetch)"

PWhite_1

8738 (Apparently) disables an optimization where rows are sorted before a Key Lookup operator. The optimization is meant to promote Sequential IO rather than the random nature of IO from Key Lookups. Note that the context in which this flag is described means that the above description may not be very precise, or even the only use of this flag.

PWhite_18

8744 `Doc2014` BOL 2014: "Disable pre-fetching for the Nested Loop operator." PWhite: "Disable prefetch (CUpdUtil::FPrefetchAllowedForDML)." Dima's article is useful to contrast NL prefetching and 8744 from NL batch sorting and 2340/9115.

920093 | Dima_1 | PWhite_1 | PWhite_2 | PWhite_19 | Connect_3

8795 PWhite: "Disable DML Request Sort (CUpdUtil::FDemandRowsSortedForPerformance)" Appears to be the counterpoint of *2332*.

PWhite_1 | PWhite_3

9115 Dima: "Disables both [NLoop Implicit Batch Sort {TF 2340} and NL Prefetching {TF 8744}], and not only on the Post Optimization, but the explicit Sort also." PWhite: "Disable prefetch (CUpdUtil::FPrefetchAllowedForDML)"

Dima_1 | PWhite_1 | Halincrest_1

Hashing and Batching

2441 Enables the use of a hash join for joins to column store indexes even when the join clause would normally be removed "during query normalization".

3146123

7359 Disables the bitmap associated with hash matching. This bitmap is used for "bit-vector filtering" and can reduce the amount of data written to TempDB during hash spills.

Dima_2

7497 Full behavior and intended purpose unknown, but the PWhite post uses it in concert with 7498 to disable "optimized bitmaps".

PWhite_4

7498 Full behavior and intended purpose unknown, but the PWhite post uses it in concert with 7497 to disable "optimized bitmaps".

PWhite_4

9347 `Doc2014` BOL 2014: "Disables batch mode for sort operator. SQL Server 2016 introduces a new batch mode sort operator that boosts performance for many analytical queries."

3172787

9349 `Doc2014` BOL 2014: "Disables batch mode for top N sort operator. SQL Server 2016 introduces a new batch mode top sort operator that boosts performance for many analytical queries."

9358 Disables batch-mode sort operations.

3171555

9453 Disables Batch Mode in Parallel Columnstore query plans. (Note that a plan using batch mode appears to require a recompile before the TF takes effect).

Niko_1

Phases and Timeouts

8671 Dima: disables the logic that prunes the memo and prevents the optimization process from stopping due to "Good Enough Plan found". Can significantly increase the amount of time, CPU, and memory used in the compilation process.

Dima_4

8677 Skips "Search 1" phase of query optimization (if applicable), and only Search 0 and Search 2 execute.

DBally_1

8757 Skip Trivial Plan optimization, essentially forcing entry into Full optimization for a query.

PWhite_5 | PWhite_6

8780 Dima: increases the "timeout" value that the optimizer sets to 3072000 transformations. Normally, the optimizer sets its internal timeout value to something based on the complexity of the query.

> **Warning:** Paul White once tweeted: "There's never a good reason to use or promote that dangerous flag"

Dima_3

8788 Dima: notes that 8788 appears to have a similar effect on the timeout as 8780, but he hasn't yet been able to determine the difference in effect between 8788 and 8780.

> **Warning:** Presumably Paul White's tweet about 8780 (above) applies here as well.

Dima_3

Miscellaneous

2329 Disables "Few Outer Rows" optimization that helps maximize parallelization of dimensional queries.

Dima_5

2335 `Doc2014` BOL 2014: "Causes SQL Server to assume a fixed amount of memory is available during query optimization. It does not limit the memory SQL Server grants to execute the query. The memory configured for SQL Server will still be used by data cache, query execution and other consumers."

KB: Causes the optimizer to generate plans that are "more conservative in terms of memory consumption when executing the query." KB describes scenario where large values of "max server memory" may lead to inefficient plans.

2413549 | PWhite_7

7470 KB: "Makes SQL Server consider internal data management memory overhead when calculating required memory for sort." Can help avoid sort spills to tempdb when the estimate is otherwise accurate.

3088480

8602 Ignore index hints that are specified in query/procedure.

Kalen_1 | SiebelPDF

8649 Strongly encourages the optimizer to generate parallel plans. (Perhaps by setting the costing for parallel exchange operators to 0.)

PWhite_5 | PWhite_8 | Machanic_1

8690 Prevents the optimizer from using "performance spools' (either table or index) in a query plan.

2962767 | CSS_2 | PWhite_9 | Connect_1 | Machanic_2 (near the end)

8692 Force optimizer to use an Eager Spool when it needs Halloween Protection

PWhite_10 | PWhite_11

8722 Disables all "other" (besides index and join) hints. This includes the OPTION clause. From Database-Wiki (but I suspect originally from a Khen book): "By running all three (8602, 8755, and 8722) flags, you can disable all hints in a query."

SQLMag_1 | Database-Wiki

8746 Whatever else it does, one effect is to disable the "rowset sharing" optimization described in the PWhite miniseries.

PWhite_11 | PWhite_12

8755 Disables all join hints.

Database-Wiki

8758 PWhite desc 1: "A [workaround to the MERGE bug described] is to apply 8758 – unfortunately this disables a number of optimisations, not just the one above, so it's not really recommended for long term use." PWhite 2: "Disable rewrite to a single operator plan (CPhyOp_StreamUpdate::PqteConvert)"

PWhite_13 | PWhite_1

8790 PWhite: "Undocumented trace flag 8790 forces a wide update plan for any data-changing query (remember that a wide update plan is always possible)."

956718 | PWhite_14 | PWhite_15

9130 Ballantyne SQLBits: "Disable non-sargable pushed predicates." Prohibits the optimizer from pushing residual predicates down into "access method" iterators (i.e. seeks and scans).

PWhite_16 | PWhite_17 | DBally_2 | Connect_2

9348 Sets a row limit (based on cardinality estimates) that controls whether a bulk insert is attempted or not (assuming conditions are met for a bulk insert). Introduced as a workaround for memory errors encountered with bulk insert.

2998301

Fixes and Past Relevance

These flags either are old and irrelevant for modern builds, appear only in CTPs, or enable a fix in a CU but are baselined in a later service pack or release.

8687 Found a reference to it (via books.google.com) in Ken Henderson's Guru's Guide to Transact-SQL, page 503: "Disables query parallelism".

8720 In SQL 2000, apparently would have the same effect as OPTION(KEEPFIXED PLAN).

Halincrest_2

9059 (Needs investigation) Turns back behavior to SQL 2000 SP3 after a SP4 installation, this allows the optimizer to choose an index seek when comparing numeric columns or numeric constants that are of different precision or scale; else would have to change schema/code.

899976

CHAPTER 11

Query Execution / Functionality

This category mashes up several smaller categories:

1. *Query Execution* –> Flags in this category affect runtime decisions made just before a query plan begins execution (or prints information about runtime behavior).
2. *Query Functionality* –> Flags in this category modify the functional definition of the syntax; most flags are quite old, and many cannot be confirmed.
3. *Query Functionality (Unable to Confirm)* –> Other public repos contain large numbers of flags that we cannot confirm. Many may be from old Sybase days, and these flags typically modify the functionality of syntax. These have been put into their own section.

The standard *Fixes and Past Relevance* section holds, as usual, flags that have trustworthy links but have ceased to be relevant for modern builds (or are fix flags whose expected utility is very short).

TODO: move 6532, 6533, and 6534 here. All are documented

Short Descriptions

Flag	Short Description
See Also	
...	
QueryExec	
646	Prints to errlog which ColStore segments were eliminated at runtime.
1504	Prints to console or errlog when an index rebuild increases its mem grant.
2466	Affects runtime DOP choice; reverts to older polling-based logic.
2467	Affects runtime DOP choice; attempts to fit query within least-loaded node.
2468	Affects runtime DOP choice; round-robin search for node that can handle query.

Continued on next page

Table 11.1 – continued from previous page

Flag	Short Description
2479	Affects runtime DOP choice; restrict query to node the connection is tied to.
2486	Enables output for query_trace_column_values Extended Event.
7525	Affects when cursors are closed upon ROLLBACK
9389	Enables dynamic memory grant for batch mode operators
...	
QueryFunc	
3640	Like SET NOCOUNT ON, eliminates sending DONE_IN_PROC messages to client.
...	
QueryFunc ?	
237	?? Use correlated sub-queries in Non-ANSI backward compatibility. ??
242	?? Provides backward compat for correlated sub-queries w/non-ANSI results. ??
243	?? Provides backward compat for nullability behavior. ??
244	?? Disables checking for allowed interim constraint violations. ??
506	?? Enforces SQL-92 standards re: null values for parm/var comparisons ??
8783	?? Allows INSERT/UPDATE/DELETE statements to honor SET ROWCOUNT ON ??
8816	?? Logs every two-digit year conversion to a four-digit year. ??
...	
Fix/PastRel	
107	Enabled the insertion of '0 into columns of type float, decimal, numeric, or real.
262	Enables a SQL 7 fix re: strings w/trailing spaces being truncated.
6530	Enables a fix for poor performance when building an index on a spatial data type.
6532	Enables a fix for poor query performance on spatial data types.
6533	Similar functionality as 6532 (performance fix for queries on spatial data).
6534	Similar functionality as 6532 (performance fix for queries on spatial data).

Query Execution

646 (Info) Prints (to SQL error log) which segments were eliminated at runtime by columnstore segment elimination. Requires 3605.

Technet_1 | Niko_1 | Niko_2 | JSack_1

1504 (Info) Prints to the console (w/3604) or the error log (w/3605; required for parallel index builds) when an index DDL command requires more memory to be granted to continue sorting rows in memory.

PWhite_1

2466 When the optimizer is choosing the runtime DOP for a parallel plan, this directs it to use logic found in "older versions" (the post doesn't say which versions) to determine which NUMA node to place the parallel plan on. This older logic relies on a polling mechanism (roughly every 1 second), and can result in race conditions where 2 parallel plans end up on the same node. The newer logic "significantly reduces" the likelihood of this happening.

CSS_1

2467 "If target MAXDOP target is less than a single node can provide and if trace flag 2467 is enabled, attempt to locate the least loaded node."

CSS_1 | CSS_2

2468 "Find the next node that can service the DOP request. Unlike full mode, the global, resource manager keeps track of the last node used. Starting from the last position, and moving to the next node, SQL Server checks for query placement opportunities. If a node can't support the request SQL Server continues advancing nodes and searching."

CSS_2

2479 When SQL Server is determining the runtime DOP for a parallel plan, this flag directs it to limit the NUMA Node placement for the query to the node that the connection is associated with.

CSS_1 | CSS_2

2486 (Info) In SQL 2016 (CTP 3.0 at least), enables output for the "query_trace_column_values" Extended Event, allowing the value of output columns from individual plan iterators to be traced.

Dima_1

7525 Affects when cursors are closed upon ROLLBACK. This flag reverts to SQL 7.0 RTM behavior. Unsure of whether this is still applicable to modern versions.

199294

9389 `Doc2014` BOL 2014: "Enables dynamic memory grant for batch mode operators...If the dynamic memory grant trace flag is enabled, a batch mode operator may ask for additional memory and avoid spilling to tempdb if additional memory is available."

Query Functionality

3640 Eliminates sending DONE_IN_PROC messages to client for each statement in stored procedure. This is similar to the session setting of SET NOCOUNT ON. (The flag gives the ability to control at a wider level without changing code).

Selvar

Query Functionality (Unable to Confirm)

Initial attempts to find online documentation that is reasonably trustworthy has not been successful.

237 Tells SQL Server to use correlated sub-queries in Non-ANSI standard backward compatibility mode

242 Provides backward compatibility for correlated subqueries where non-ANSI-standard results are desired.

243 Provides backward compatibility for nullability behavior. When set, SQL Server has the same nullability violation behavior as that of a ver 4.2:

1. Processing of the entire batch is terminated if the nullability error (inserting NULL into a NOT NULL field) can be detected at compile time.
2. Processing of offending row is skipped, but the command continues if the nullability violation is detected at run time.

Behavior of SQL Server is now more consistent because nullability checks are made at run time and a nullability violation results in the command terminating and the batch or transaction process continuing.

244 Disables checking for allowed interim constraint violations. By default, SQL Server checks for and allows interim constraint violations. An interim constraint violation is caused by a change that removes the violation such that the constraint is met, all within a single statement and transaction. SQL Server checks for interim constraint violations for self-referencing DELETE statements, INSERT, and multi-row UPDATE statements. This checking requires more work tables. With this trace flag you can disallow interim constraint violations, thus requiring fewer work tables.

506 Enforces SQL-92 standards regarding null values for comparisons between variables and parameters. Any comparison of variables and parameters that contain a NULL always results in a NULL.

8783 Allows DELETE, INSERT, and UPDATE statements to honor the SET ROWCOUNT ON setting when enabled.

8816 Logs every two-digit year conversion to a four-digit year.

Fixes and Past Relevance

These flags either are old and irrelevant for modern builds, appear only in CTPs, or enable a fix in a CU but are baselined in a later service pack or release.

107 Enabled the insertion of '0 into columns of type float, decimal, numeric, or real.

160732

262 Enables a fix in SQL 7.0 to correct a problem with strings w/trailing spaces being truncated even when ANSI PADDING was on.

891116

6530 Enables a fix for poor performance when building an index on a spatial data type.

2896720

6532 `Doc2014` BOL 2014: "Enables performance improvement of query operations with spatial data types in SQL Server 2012 and SQL Server 2014. The performance gain will vary, depending on the configuration, the types of queries, and the objects. Note: Beginning with SQL Server 2016 this behavior is controlled by the engine and trace flag 6532 has no effect."

3107399

6533 `Doc2014` BOL 2014 has an identical description for this flag as for *6532*.

3107399

6534 `Doc2014` BOL 2014 has an identical description for this flag as for *6532*.

3107399

CHAPTER 12

Plan Caching

Most of the flags affect the size of the plan cache, or how certain types of plans (e.g. zero-cost, ad-hoc) are cached. This is a very small category and may be combined with another at a later time.

Short Descriptions

Flag	Short Description
See Also	These flags significantly affect the retention of cached plans.
205	Reports to error log when recompile occurs due to auto-update of stats.
2389	Enables optimizer tracking of ascending keys and subsequent recompilation behavior.
2390	Similar to 2390 and the ascending key problem.
2453	Applies recompile thresholds for temp tables to table variables
4139	Enables "quick stats" histogram amendments even if key col isn't branded ascending.
* *2371*	Lowers relative threshold for stats-based recompilation as table gets larger
...	
Func Toggles	
174	Increases plan cache bucket count from 40k to 160k
253	(May) prevent ad-hoc query plans from being cached.
2861	Instructs SQL Server to keep zero cost plans in cache
8032	Reverts the cache limit parameters to the SQL Server 2005 RTM settings.
...	
Fix/PastRel	
2880	Enables an upper bound for the plan cache in SQL 2000 32-bit.
2881	Turns off an upper bound on the plan cache in SQL 2000 64-bit

Functionality Toggles

174 `Doc2014` BOL 2014: "Increases the SQL Server Database Engine plan cache bucket count from 40,009 to 160,001 on 64-bit systems." Can relieve SOS_CACHESTORE spinlock contention though at the cost of increasing the amount of memory allowed for the plan cache.

3026083

253 SSC repository: "Prevents ad-hoc query plans to stay in cache." Even though many "three-digit" query plan numbers don't seem to be relevant any more, the first SO hit indicates that this flag still changes functionality in the manner described by the SSC definition. As a side note, Martin Smith's comment in the second Stack Overflow hit about DAC queries never being cached is interesting.

StackOverflow_1 | DBAse_1

2861 KB: "Instructs SQL Server to keep zero cost plans in cache, which SQL Server would typically not cache (such as simple ad-hoc queries, set statements, commit transaction and others)."

325607 | SolarWindsDPA | SQLMag_1 | DBAse_2

8032 `Doc2008R2` BOL 2014: "Reverts the cache limit parameters to the SQL Server 2005 RTM setting which in general allows caches to be larger. Use this setting when frequently reused cache entries do not fit into the cache and when the optimize for ad hoc workloads Server Configuration Option has failed to resolve the problem with plan cache. WARNING: Trace flag 8032 can cause poor performance if large caches make less memory available for other memory consumers, such as the buffer pool."

Banerjee: Can't be used as a startup flag; @sql_handle tweet indicates it can ONLY be used as a startup flag in SQL 2014 (and I would guess in 2012 as well, since that's when the SQLOS mem rearchitecture occurred). Another @sql_handle tweet indicates it also increases size of log pool cache.

Fixes and Past Relevance

These flags either are old and irrelevant for modern builds, appear only in CTPs, or enable a fix in a CU but are baselined in a later service pack or release.

2880 Enables an upper bound for the plan cache in SQL 2000 32-bit. (The upper bound is by default disabled in 32-bit SQL 2000 and enabled in 64-bit SQL 2000).

891707

2881 Turns off an an upper bound on how large the plan cache can get. (Upper bound introduced in 64-bit SQL 2000 to solve a prob with the plan cache and adhoc sql). Does the flag still do anything?

891707

CHAPTER 13

Databases and Files

Flags in this category fall into the below groupings:

1. *Allocation* –> Flags that modify how pages in MDF/NDF files are allocated to individual objects.
2. *Other* –> Flags not related to page allocation.
3. *Fixes and Past Relevance* –> Fix flags and flags which are no longer relevant.

See also: 3449 and 8903 ([Disk and Network IO](https://github.com/AaronMorelli/SQLServerTraceFlags/blob/master/Categories/DiskNetworkIO.md))

(TODO: ensure 1802 is present in the See Also for Security) (TODO: find other entry for 9394 and place an in-line link to its other home. Ditto for the other section) (TODO: 10207 should be moved to the corrupt Checkdb section and placed here in the see also section) TODO: is 9929 in multiple locations? e.g. the special features one? TODO: should 1482 move to a t-log focused page? TODO: 2330 should have a reference in one of the optimization pages

Short Descriptions

Flag	Short Description
See Also	
205	Reports to error log when recompile occurs due to auto-update of stats.
...	
Allocation	
670	(or 671) Disables deferred deallocation.
1106	Creates a new ring buffer to track TempDB allocations.
* *1117*	When one files grows, all files in filegroup grow with it.
* *1118*	Eliminates most single page allocations.
1165	Prints calculations used for proportional fill algorithm.

Continued on next page

Table 13.1 – continued from previous page

Flag	Short Description
1180	Changes the way text/image data is allocated
1197	Prevents use of worktable cache for certain TempDB allocations.
1806	Disables instant file initialization
...	
Other	
272	Reverts identity column generation behavior to pre-SQL 2012 behavior.
647	Skips a new-in-2012 data check that lengthens ALTER TABLE durations.
1140	Allows reversion to an older algorithm for mixed page allocs.
1482	Prints info on a variety of t-log management operations.
1808	Ignores auto-closing DBs even if auto-close property is set.
1816	Provides more details around IO errors for stretch database.
2330	Stops data collection for sys.db_index_usage_stats.
2548	Causes DBCC SHRINK and other options to skip LOB_COMPACTION.
3861	Allows DB startup code to move system tables to primary FG.
9851	Disables Hekaton auto-merge process for checkpoint files.
9929	Reduces Hekaton checkpoint files to 1 MB each.
10204	Disables merge/recompress of smaller columnstore delta rowgroups.
10207	Disables skipping of corrupt segments for columnstore scans.
...	
Fix/PastRel	
1802	Workaround for permissions change upon DB detach
1807	Allows creation of DB file on UNC paths
9394	Enables fix for AV re: Japanese characters
10202	Enabled new column store DMV in pre-release SQL version.

Allocation

670 (or 671) CSS: Disables deferred deallocation. But note Paul White's comment on the post! The flag number may actually be 671.

CSS_1

1106 (Info) Creates a new RB in sys.dm_os_ring_buffers that tracks allocations made in TempDB.

947204 | BobWard_Pass2011 | Arvind_1

1117 `Doc2014` BOL 2014: "When a file in the filegroup meets the autogrow threshold, all files in the filegroup grow.

Note: Beginning with SQL Server 2016 this behavior is controlled by the AUTOGROW_SINGLE_FILE and AUTOGROW_ALL_FILES option of ALTER DATABASE, and trace flag 1117 has no affect."

This flag is commonly associated with tempdb but applies to all databases when on. The flag is typically used to ensure that all files grow evenly to maintain a well balanced proportional-fill allocation algorithm. Nacho gives a very special/rare edge case for sysfiles1. Chris Adkin has some interesting screenshots on its effect under certain workloads. (PRand_1 doesn't reference this flag but its info is highly relevant to this topic.)

ALTER DATABASE file and filegroup options | BobWard_Pass2011 | PRand_4 | Nacho_2 | CAdkin_2 | SQLArticlesDotCom

1118 `Doc2014` BOL 2014: "Removes most single page allocations on the server, reducing contention on the SGAM page. When a new object is created, by default, the first eight pages are allocated from different extents (mixed

extents). Afterwards, when more pages are needed, those are allocated from that same extent (uniform extent). The SGAM page is used to track these mixed extents, so can quickly become a bottleneck when numerous mixed page allocations are occurring. This trace flag allocates all eight pages from the same extent when creating new objects, minimizing the need to scan the SGAM page. For more information, see this Microsoft Support article.

Note: Beginning with SQL Server 2016 this behavior is controlled by the SET MIXED_PAGE_ALLOCATION option of ALTER DATABASE, and trace flag 1118 has no affect."

ALTER DATABASE SET Options | 328551 | 837938 | 936185 | 2154845 | CSS_3 | CSS_4 | PRand_5 | CAdkin_2

1140 Allows reversion to an older, more aggressive form of the mixed-page-allocation algorithm. The flag was introduced as a workaround for a bug in 2005SP2/SP3 and SQL 2008 where mixed page allocations climb continually in tempdb for workloads that use tempdb extremely heavily. That behavior was due to a change in the way that mixed-page allocations are done. KB has a great description of both the "old" and "new" way that free pages are found for a mixed-page allocation to be performed.

2000471

1165 (Info) Outputs the recalculated #'s (every 8192 allocations) for the proportional fill algorithm when multiple files are present. Requires TF 3605, output goes to SQL error log.

BobWard_Pass2011 | PRand_1

1180 (Very old, may not be functional) KB notes that after a SQL 7.0 fix is installed, this flag will cause text/image data to be placed in free pages in partially-allocated extents; w/o the flag, text/image data is placed in newly-allocated extents until the file size limit is reached; only then will partially-allocated extents be used for new data.

272220

1197 Bob Ward uses to prevent allocation of TempDB pages (by Work Tables) from being pulled from a worktable cache (see around 1:25:00). The (very old) KB references for use w/1180 in reclaiming space from inefficiently-stored text/image data.

BobWard_Pass2011 | 324432

1806 Disables instant file initialization.

2574695 | PFE_1 | PRand_2

Other

272 Connect: "In SQL Server 2012 the implementation of the identity property has been changed to accommodate investments into other features. In previous versions of SQL Server the tracking of identity generation relied on transaction log records for each identity value generated. In SQL Server 2012 we generate identity values in batches and log only the max value of the batch. This reduces the amount and frequency of information written to the transaction log improving insert scalability. If you require the same identity generation semantics as previous versions of SQL Server there are two options available:

- Use trace flag 272. This will cause a log record to be generated for each generated identity value. The performance of identity generation may be impacted by turning on this trace flag.
- Use a sequence generator with the NO CACHE setting. This will cause a log record to be generated for each generated sequence value. Note that the performance of sequence value generation may be impacted by using NO CACHE."

Later in the Connect discussion, one commenter notes that when adding the TF as a startup flag, the flag only appears to work when using the "lowercase t" syntax rather than the more common "uppercase T" syntax.

Connect_1

647 Avoids a new-in-SQL 2012 data check (done when adding a column to a table) that can cause ALTER TABLE... ADD <column> operations to take a very long time. The KB has a useful query for determining the row size for a table.

2986423

1482 (Info) Prints info to the Error Log (3605 not necess.) for a variety of transaction log operations, including when MinLSN value is reset, when a VLF is formatted, etc. (First discovered in Bob Ward's PASS 2014 talk on SQL Server IO, and then tested for myself.)

<links needed>

1808 Directs SQL Server to ignore auto-closing databases even if the Auto-close property is set to ON. Must be set globally. Present in 2005+.

Nacho_1

1816 (Info) Bob Ward briefly references this flag in his PASS2014 IO talk, saying it "could provide more details around errors" that occur with IO done to SQL data files in Azure Storage (stretch/http IO, I think he means).

<links needed>

2330 Stops the collection of statistics for sys.db_index_usage_stats. CAdkin: also disables sys.dm_db_missing_index_group_stats, and thus is useful when seeing high waits on the OPT_IDX_STATS spinlock.

2003031 | PRand_3 | BrentOzar_1 | CAdkin_1

2548 "SQL 2005 has a –T2548 dbcc tracon(-1, 2548) that allows shrink* and other LOB_COMPACTION actions to be skipped. Enabling this returns shrink* behavior to that similar to SQL 2000."

CSS_2

3861 Allows the DB startup code to move system tables to the primary filegroup. Introduced for a bug in SQL 2014 upgrade process, where system tables could be created in a secondary filegroup (if that FG was the default).

3003760

9851 Disables Hekaton's auto-merge process; if this flag is enabled, the various merge-related procedures will need to be called manually. First seen in a Sunil Agarwal session at PASS 2014, also present in Kalen Delaney's book on Hekaton.

<links needed>

9929 Enables an update that reduces the "disk footprint [of In-Memory OLTP] by reducing the In-Memory checkpoint files to 1 MB (megabytes) each."

3147012

10204 `Doc2016` BOL 2016: "Disables merge/recompress during columnstore index reorganization. In SQL Server 2016, when a columnstore index is reorganized, there is new functionality to automatically merge any small compressed rowgroups into larger compressed rowgroups, as well as recompressing any rowgroups that have a large number of deleted rows."

10207 When a CCI is corrupt, allows a scan to skip corrupt segments and suppress errors 5288 and 5289, thus enabling the copy-out of data in a corrupt CCI.

3067257 | RelSvcs_1

Fixes and Past Relevance

These flags either are old and irrelevant for modern builds, appear only in CTPs, or enable a fix in a CU but are baselined in a later service pack or release.

1802 Workaround for: "after you detach a Microsoft SQL Server 2005 database that resides on network- attached storage, you cannot reattach the SQL Server database... This problem occurs because SQL Server 2005 resets the file permissions when the database is detached. When you try to reattach the database, it cannot be attached because of limited share permissions."

It sounds like this flag disables functionality in changing permissions on database files after the DB is detached, thus has security implications.

922804 | StorEng_1 (comments, though Kevin likely means 1807)

1807 Allows the creation of a database file on UNC paths, and is a workaround for errors 5105 and 5110. The KB describes MSFT policy towards DBs on network locations.

304261

9394 (9394 is either doing double-duty or there's a typo. See other entry for 9394) Apparently enables a fix for an access violation when a table with Japanese characters has an indexed changed.

3142595

10202 Sunil Agarwal PASS 2014 demo script: enables new DMV named sys.dm_db_column_store_row_group_physical_stats. DMV was not in 2014 at the time of the demo, thus appears to be in a future (or internal) version of SQL Server.

CHAPTER 14

Locking and Waiting

Flags in this category fall into the below groupings:

1. *Informational* –> Flags that return info about locks and deadlocks.
2. *Functionality Toggles* –> Flags that modify lock escalation or other locking behaviors.
3. *Fixes and Past Relevance* –> Fix flags and flags which are no longer relevant.

See also:

1229 prob needs a See Also in the scheduler section.

Is 8755 already referenced in the query optimization section?

Short Descriptions

Flag	Short Description
See Also	
205	Reports to error log when recompile occurs due to auto-update of stats.
...	
Info	
611	Records lock escalations in the SQL error log.
1200	Prints detailed lock info for every lock request/release.
* *1204*	Prints detailed info upon every deadlock occurrence.
1205	Prints info to SQL error log every time a deadlock search begins.
1206	Complements 1204 by printing other locks held by deadlock parties.
1208	Prints host name/program name [by clients in a deadlock?]
1222	Similar to 1204, but with more info and in an XML format.
8001	Adds more wait types to sys.dm_os_wait_stats
8050	Removes "optional" wait types from sys.dm_os_wait_stats.
...	
Func	
* *1211*	Disables lock escalations due to memory pressure.
* *1224*	Disables lock escalations based on number of locks.
1229	Disables lock partitioning among schedulers.
8755	Disables any locking hints in query text.
...	
Fix/PastRel	
1216	Complicated; see below
1236	Reduces LOCK_HASH spinlock contention on older builds
2456	Relieves RESOURCE_SEMAPHORE_MUTEX contention in SQL 2005.

Informational

611 (Info) Records each lock escalation, in the form: "Escalated locks - Reason: LOCK_THRESHOLD, Mode: S, Granularity: TABLE, Table: 222623836, HoBt: 150:256, HoBt Lock Count: 6248, Escalated Lock Count: 6249, Line Number: 1, Start Offset: 0, SQL Statement: select count(*) from dbo.BigTable"

Aaron: Confirmed still in SQL 2014.

SSWUG_1

1200 Prints detailed lock info as every request for a lock is made (the process ID and type of lock requested).

169960 | StorEng_1 (comments)

1204 `Doc2005` **(Info)** BOL 2014: "Returns the resources and types of locks participating in a deadlock and also the current command affected.".

937950 notes a specific type of error where 1204 will not help with deadlocks.

832524 | 937950

1205 (Info) Prints information to the SQL error log every time a deadlock search starts, whether or not a deadlock is found.

832524

1206 (Info) Used to complement flag 1204 by displaying other locks held by deadlock parties.

169960

1208 (Info) KB: "Prints the host name and program name supplied by the client. This can help identify a client involved in a deadlock, assuming the client specifies a unique value for each connection."

169960

1222 `Doc2005` (Info) BOL 2014: "Returns the resources and types of locks that are participating in a deadlock and also the current command affected, in an XML format that does not comply with any XSD schema."

8001 (Info) Khen2005, p2: "For SQL Server 2005, the SQL Server product team opted not to include [in sys.dm_os_wait_stats] some wait types that fall under one of the following three categories:

- Wait types that are never used in SQL Server 2005; note that some wait types not excluded are also never used.
- Wait types that can occur only at times when they do not affect user activity, such as during initial server startup and shutdown, and are not visible to users.
- Wait types that are innocuous but have caused concern among users because of their high occurrence or duration

The complete list of wait types is available by enabling trace flag 8001. The only effect of this trace flag is to force sys.dm_os_wait_stats to display all wait types."

8050 (Info) Causes "optional" wait types (see the CSS article) to be excluded when querying sys.dm_os_wait_stats.

CSS_2

Functionality Toggles

1211 `Doc2005` BOL 2014: "Disables lock escalation based on memory pressure, or based on number of locks. The SQL Server Database Engine will not escalate row or page locks to table locks.

Using this trace flag can generate excessive numbers of locks. This can slow the performance of the Database Engine, or cause 1204 errors (unable to allocate lock resource) because of insufficient memory.

If both trace flag 1211 and 1224 are set, 1211 takes precedence over 1224. However, because trace flag 1211 prevents escalation in every case, even under memory pressure, we recommend that you use 1224. This helps avoid "out-of-locks" errors when many locks are being used."

PRand_1

1224 `Doc2005` BOL 2014: "Disables lock escalation based on the number of locks. However, memory pressure can still activate lock escalation. The Database Engine escalates row or page locks to table (or partition) locks if the amount of memory used by lock objects exceeds one of the following conditions:

- Forty percent of the memory that is used by Database Engine. This is applicable only when the locks parameter of sp_configure is set to 0.
- Forty percent of the lock memory that is configured by using the locks parameter of sp_configure.

If both trace flag 1211 and 1224 are set, 1211 takes precedence over 1224. However, because trace flag 1211 prevents escalation in every case, even under memory pressure, we recommend that you use 1224. This helps avoid "out-of-locks" errors when many locks are being used.

Note: Lock escalation to the table- or HoBT-level granularity can also be controlled by using the LOCK_ESCALATION option of the ALTER TABLE statement."

PRand_1

1229 Disables lock partitioning (among schedulers).

CSS_1

8755 Disables any locking hints and permits the optimizer/query execution engine to select the appropriate lock behavior.

SQLMag_1

Fixes and Past Relevance

These flags either are old and irrelevant for modern builds, appear only in CTPs, or enable a fix in a CU but are baselined in a later service pack or release.

1216 (Info?) 319892: mentions 1216 in passing as a flag in SQL 2000 that is associated with deadlock output. The mention occurs only to distance TF1216 on SQL 2000 from TF 1216 on SQL 7.0 and TF 1261 on SQL 2000, which both have a different purpose than 1216 on SQL 2000. Prob needs to be tested on modern versions.

319892

1236 (On by default in SQL 2014 SP1+ and SQL 2012 SP3) Partitions the DATABASE lock type to help reduce contention on internal locking structures symptomized by LOCK_HASH spinlock contention.

2926217

2456 Relieves RESOURCE_SEMAPHORE_MUTEX contention, which may be primarily due to a bug in SQL 2005.

956375

CHAPTER 15

Transactions, Logging, and Recovery

All flags in this category are related to transactional behavior or transaction records being written to disk (to the transaction log).

See also: 9592 and 9567 High Availability/Distributed Technologies https://github.com/AaronMorelli/SQLServerTraceFlags/blob/master/Categories/HADRDist.md

Short Descriptions

Flag	Short Description
See Also	
205	Reports to error log when recompile occurs due to auto-update of stats.
...	
Flags	
* *610*	Enables minimal logging under certain scenarios.
2537	Enables fn_dblog() to read the inactive portion of the t-log.
3412	Reports when each tran is rolled forward or back.
8599	Allows the use of save points within distributed trans.
9038	Disables the multi-threaded log writer in SQL 2016.
9909	Forces single-threaded ALTER TABLE.
...	
Fix/PastRel	
3924	Enables a fix for "XA" trans w/in JDBC connections.
9024	Partitions a log pool memory object by memory node.

Flags

610 Enables minimal-logging under certain scenarios. Connect item indicate TF 610 soon may be a query hint.

Data Loading Perf Guide | StorEng_1 | Connect_1

2537 Enables fn_dblog() to read all of the t-log, not just the active portion of the log. Argenis Tweet: fn_dblog + 2537 is a way to check your t-log file for corruption.

PRand_1 | PRand_2

3412 Reports when each transaction is rolled forward or back, except for large transactions. (Needs verification for whether still current)

170115

8599 (Very old, needs verification.) Allows the use of a save-point within a distributed transaction. Note that one reference to the flag in KB295027 is written incorrectly as 8588.

295027 | 903742

9038 CAdkin Tweet: "If you are using low-latency storage it appears that you can also cause high spins on LOGFLUSHQ in SQL 2016 with the in-memory engine and full durability. The multithreaded log writer is the cause of this and can be turned off via TF 9038."

9909 Turns off a logging optimization when an ALTER TABLE operation on a memory-optimized table would normally be done in parallel. This avoids the possibility of data loss b/c of a bug in parallel ALTER TABLE.

3174963

Fixes and Past Relevance

These flags either are old and irrelevant for modern builds, appear only in CTPs, or enable a fix in a CU but are baselined in a later service pack or release.

3924 Enables a fix where "XA" transactions started within a JDBC-connected Java app are not closed properly and stay open indefinitely.

3145492

9024 (On by default as of SQL 2014 SP1+) Enables a fix in SQL 2012/2014 that partitions (by memory node) a "pointer to a memory object" (PMO) to the log pool. The KB references this flag specifically in the context of Availability Groups, though the flag appears to deal with the design of the "logpool". Note that 8048 is also potentially applicable here, as it will further partition the PMO to a by-CPU partitioning scheme. (See the KB).

2809338

CHAPTER 16

Backup and Restore

Flags in this category fall into the below groupings:

1. *Informational* –> Flags that return info about the BACKUP or RESTORE process.
2. *Functionality Toggles* –> Flags that modify how BACKUP or RESTORE behavior.
3. *Fixes and Past Relevance* –> Fix flags and flags which are no longer relevant.

See also:

TODO: 3028 may need to be moved to a fix flag section, ditto for 3057, 3111, 3117 TODO: 3607 should at least be referenced in the TranLog Recovery section, and in the CheckDB Corruption section. TODO: 9958 should at least be referenced in the TranLog Recovery section TODO: does 3222 belong here, or as a See Also anywhere else?

Short Descriptions

Flag	Short Description
See Also	
205	Reports to error log when recompile occurs due to auto-update of stats.
...	
Info	
3001	Suppresses logging to msdb.dbo.backuphistory
3004	Prints info (e.g. file initialization) re: BACKUP/RESTORE operations
3014	Prints info about config options used during BACKUP.
3210	Prints info about Async Disk Pool thread collision and wait times.
3212	Prints “backup stats” to the SQL log.
3213	Prints info about BACKUP param values used, e.g. Max Transfer Size

Continued on next page

Table 16.1 – continued from previous page

Flag	Short Description
3216	Prints info about RESTORE internals.
* *3226*	Suppresses BACKUP messages to the error log.
3400	Appears to print out various info about RESTORE process.
...	
Func	
3034	Overrides server default and forces backup compression
3035	Overrides server default and avoids backup compression
3039	Allows VDI backups to be affected by the server default compression setting.
3042	Causes BACKUP to allocate size of .bak file as needed instead of up-front.
3205	Disables hardware compression for tape drivers during BACKUP or RESTORE.
3222	Disables read-ahead used during recovery operation during roll-forward.
3422	Enables auditing of t-log records during rollback or log recovery.
3607	Lets SQL Server open the master database w/out running recovery on it.
3608	SQL Server only starts up master, not any other DB.
3609	Skips creation of tempdb at startup.
3660	Forces database recovery at SQL Server startup even for auto-close DBs.
...	
Fix/PastRel	
3023	Enables CHECKSUM option as default for all BACKUPs.
3028	Enables a hotfix for problems backing up to tape.
3031	Turns NO_LOG and TRUNCATE_ONLY into checkpoints.
3057	Allows log backups w/sector size 512 to be restore on sector size 4096.
3101	Causes RESTORE to bypass a CDC upgrade operation.
3111	Enables a fix for backing up very large T-log files.
3117	Enables a fix when restoring from a file/FG-based snapshot backup.
3231	Turns NO_LOG/TRUNCATE_ONLY into no-ops/log clears based on recov mode.
9109	Enables a workaround for restoring a DB w/query notifs and NEW_BROKER.
9958	Enables a fix for a bug restoring a t-log with hekaton records.

Informational

3001 (Info) Erland: "suppresses logging to msdb.backuphistory...undocumented with all that means. On the other hand, I got [it] from a Microsoft engineer who said it was OK to share them."

Erland_1

3004 (Info) Prints trace info to the SQL error log about RESTORE (and BACKUP?) operations and can be used to view file initialization; use with 3605 to direct to error log

CSS_1_ | SQLPFE_1 | SQLMunkee

3014 (Info) Prints info to the error log about config option values chosen during the BACKUP command. In the CSS blog post, used with 3213 (not sure how they are different; more testing is needed).

CSS_2

3210 (Info) Bob Ward PASS 2014 IO talk: prints information about "collisions and wait times" that occur between the various "Asynchronous Disk Pool" threads during BACKUP (what about RESTORE?) operations.

<links needed>

3212 (Info) Prints "Backup stats" to the SQL log.

Nacho_1

3213 (Info) Prints info about BACKUP parameter values used, especially regarding Buffer size/number and Max Transfer size.

CSS_2 | CSS_3

3216 (Info) Prints info about RESTORE internals. Only seems to print to the error log (TF 3605 is required). Not able to find an official link.

JamesSQL

3226 `Doc2008` (Info) BOL 2014: "By default, every successful backup operation adds an entry in the SQL Server error log and in the system event log. If you create very frequent log backups, these success messages accumulate quickly, resulting in huge error logs in which finding other messages is problematic.

With this trace flag, you can suppress these log entries. This is useful if you are running frequent log backups and if none of your scripts depend on those entries."

StorEng_1 | PRand_1

3400 (Info) Connect: appears (based on context) to print information re: the RESTORE process. BWard PASS 2014 IO talk: explained as printing out "checkpoint pages/sec" (to the Error Log, presumably)

Connect_1

Functionality Toggles

3034 (is this just for VDI?) overrides the server default, and thus always forces backup compression unless the backup command had the no_compression clause explicitly present.

Nacho_2

3035 (is this just for VDI?) overrides the server default to always avoid compression, unless the backup command explicitly uses the compression clause. If both 3034 and 3035 are enabled, 3035 takes precedence.

Nacho_2

3039 As long as the SQL edition supports backup compression, this will allow VDI backups to be affected by the default compression setting just as non-VDI BACKUP commands are affected.

Nacho_2

3042 `Doc2012` BOL 2014: "Bypasses the default backup compression pre-allocation algorithm to allow the backup file to grow only as needed to reach its final size. This trace flag is useful if you need to save on space by allocating only the actual size required for the compressed backup. Using this trace flag might cause a slight performance penalty (a possible increase in the duration of the backup operation)."

The KB article discusses the algorithm used to estimate space when the TF is NOT on.

2001026 | CSS_4

3205 `Doc2005` BOL 2014: "By default, if a tape drive supports hardware compression, either the DUMP or BACKUP statement uses it. With this trace flag, you can disable hardware compression for tape drivers. This is useful when you want to exchange tapes with other sites or tape drives that do not support compression."

3222 Disables the read ahead that is used by the recovery operation during roll forward operations.

268081

3422 PRand: "Cause auditing of transaction log records as they're read (during transaction rollback or log recovery). This is useful because there is no equivalent to page checksums for transaction log records and so no way to detect whether log records are being corrupted." [There is a CPU hit for turning this on].

IO Basics, chapter 2 | PRand_2

3607 Khen2005, page 80: lets SQL open master w/out running recovery on it. Other sources say that SQL doesn't try to "start up" master. The differences in wording may not be important.

Nacho_1

3608 `Doc2008` BOL 2014: "Prevents SQL Server from automatically starting and recovering any database except the master database. If activities that require tempdb are initiated, then model is recovered and tempdb is created. User databases will be started and recovered when accessed. Some features, such as snapshot isolation and read committed snapshot, might not work."

Nacho_1 | Nacho_3

3609 Skips the creation of the tempdb database at startup. Use this trace flag if the device or devices on which tempdb resides are problematic or problems exist in the model database.

PRand_3 | Nacho_1

3660 W/o this flag, for DBs that have Auto_Close=true and for DBs on Express Edition, DB recovery is normally deferred until first user access when SQL starts up. This TF forces DB recovery to always run (well, only for DBs that actually need recovery done) at SQL Server startup.

Nacho_4

Fixes and Past Relevance

These flags either are old and irrelevant for modern builds, appear only in CTPs, or enable a fix in a CU but are baselined in a later service pack or release.

3023 `Doc2014` BOL 2014: "Enables CHECKSUM option as default for BACKUP command. Note: Beginning with SQL Server 2014 this behavior is controlled by setting the backup checksum default configuration option."

2656988

3028 Enables a hotfix for a problem encountered when backing up to tape with specific backup options.

940379

3031 Will turn the NO_LOG and TRUNCATE_ONLY options into checkpoints in all recovery modes (applicable to SQL 2005).

PRand_4

3057 Enables a hotfix change that allows a log backup that was taken on a t-log file hosted on a drive with "Bytes per physical sector"=512 to be restored onto a log file/drive that has "Bytes per physical sector"=4096.

2987585

3101 Causes the RESTORE process to bypass a CDC upgrade operation that can cause slow RESTORE operations under certain conditions.

2567366

3111 "FIX: Backup or Restore Using Large Transaction Logs May Return Error 3241" Causes LogMgr::ValidateBackedupBlock to be skipped during backup and restore operations, allowing backups of very large T-logs to succeed.

297104

3117 Enables a fix in 2005 when restoring from a file- or filegroup-based snapshot backup. 3117 causes the restore process to use an approach found in SQL 2000 rather than 2005's logic.

915385

3231 SQL 2000/2005 - Will turn the NO_LOG and TRUNCATE_ONLY options into no-ops in FULL/BULK_LOGGED recovery mode, and will clear the log in SIMPLE recovery mode.

PRand_4 | KTripp_1

9109 Used to workaround a problem with query notifications and restoring a DB with the NEW_BROKER option enabled.

2483090

9958 Enables a fix that allows the restore of a Hekaton transaction log backup when a certain bug is hit.

3171002

CHAPTER 17

CHECKDB and Corruption

Flags in this category fall into the below groupings:

1. *CHECKDB* –> Flags that modify CHECKDB behavior or return additional info.
2. *Engine Detection* –> Flags that enable additional checking for corruption by core engine activities.
3. *Fixes and Past Relevance* –> Fix flags and flags which are no longer relevant.

See also:

TODO: Strongly consider combining this and the BackupRestore section somehow

TODO: Reconsider where 2509 and 2514 go. (Have to do with file/page internals, but activated by DBCC CHECKDB)

Short Descriptions

Flag	Short Description
See Also	
205	Reports to error log when recompile occurs due to auto-update of stats.
...	
CHECKDB	
2508	(speculative) Disables parallel non-clustered idx checks in CHECKTABLE.
2514	Causes DBCC CHECKTABLE to return total count of ghost records.
* *2528*	Disables parallel checking of objects by DBCC CHECK* commands.
2529	Appears to print memory usage by DBCC CHECKDB.
2549	Causes CHECKDB to assume each file is on a unique LUN.
2558	Disables integration between CHECKDB and Watson.
* *2562*	Runs all DB objects in a single CHECKDB batch.
2566	Disables DATA_PURITY checks in CHECKDB.
...	
Engine	
806	Enables "DBCC-style" auditing on each page read.
815	Enables latch enforcement to help catch memory scribblers.
818	Adds auditing for disk write IOs.
813_	Protects unchanged buffer pool changes from memory corruptions.
...	
Fix/PastRel	
2509	Possibly the SQL 2000 equivalent to TF 2514 above.

CheckDB

2508 TODO: experiment to see if this has any kernel of truth, may just be a typo of 2528: Social.Technet: "Disables parallel non-clustered index checking for DBCC CHECKTABLE"

Social.Technet

2514 (Info) Used with DBCC CHECKTABLE to see the total count of ghost records in a table.

Argenis_1

2528 ‘‘Doc2005‘ BOL 2014: "Disables parallel checking of objects by DBCC CHECKDB, DBCC CHECKFILEGROUP, and DBCC CHECKTABLE. By default, the degree of parallelism is automatically determined by the query processor. The maximum degree of parallelism is configured just like that of parallel queries. For more information, see Configure the max degree of parallelism Server Configuration Option.

Parallel DBCC should typically be left enabled. For DBCC CHECKDB, the query processor reevaluates and automatically adjusts parallelism with each table or batch of tables checked. Sometimes, checking may start when the server is almost idle. An administrator who knows that the load will increase before checking is complete may want to manually decrease or disable parallelism.

Disabling parallel checking of DBCC can cause DBCC to take much longer to complete and if DBCC is run with the TABLOCK feature enabled and parallelism set off, tables may be locked for longer periods of time."

PRand_2

2529 (Info?) Full functionality unknown, but appears to print memory usage for CHECKDB (memory that appears to be allocated via an "IMemObj" interface) at the very beginning and very end of CHECKDB output.

SQLService.se

2549 `Doc2014` BOL 2014: "Runs the DBCC CHECKDB command assuming each database file is on a unique disk drive. DBCC CHECKDB command builds an internal list of pages to read per unique disk drive across all database files. This logic determines unique disk drives based on the drive letter of the physical file name of each file.

Note: Do not use this trace flag unless you know that each file is based on a unique physical disk.

Note: Although this trace flag improve the performance of the DBCC CHECKDB commands which target usage of the PHYSICAL_ONLY option, some users may not see any improvement in performance. While this trace flag improves disk I/O resources usage, the underlying performance of disk resources may limit the overall performance of the DBCC CHECKDB command."

2634571 | BobWard_1 | CSS_2 | Bertrand_1 | PRand_3 | Connect_1 (problems w/SQL 2014)

2558 Disables integration between CHECKDB and Watson.

CSS_3

2562 `Doc2014` BOL 2014: "Runs the DBCC CHECKDB command in a single "batch" regardless of the number of indexes in the database. By default, the DBCC CHECKDB command tries to minimize tempdb resources by limiting the number of indexes or "facts" that it generates by using a "batches" concept. This trace flag forces all processing into one batch.

One effect of using this trace flag is that the space requirements for tempdb may increase. Tempdb may grow to as much as 5% or more of the user database that is being processed by the DBCC CHECKDB command.

Note: Although this trace flag improve the performance of the DBCC CHECKDB commands which target usage of the PHYSICAL_ONLY option, some users may not see any improvement in performance. While this trace flag improves disk I/O resources usage, the underlying performance of disk resources may limit the overall performance of the DBCC CHECKDB command."

2634571 | BobWard_1 | CSS_2 | Bertrand_1 | PRand_3 | PRand_4

2566 Disables DATA_PURITY checks (in CHECKDB) and was released as a workaround for several problems in x64 instances (as described in the KB articles).

945770 | 2888996 | Bertrand_1

Engine Detection

806 PRand: "Causes 'DBCC-style' page auditing to be performed whenever a database page is read into the buffer pool. This is useful to catch cases where pages are being corrupted in memory and then written out to disk with a new page checksum. When they're read back in the checksum will look correct, but the page is corrupt (because of the previous memory corruption). This page auditing goes someway to catching this - especially on non-Enterprise Edition systems that don't have the 'checksum sniffer'." [Paul notes there will be a CPU hit if you turn this on].

841776 | IOBasics Chapter 2 | PRand_1

815 IO Basics: "To help detect unwanted changes to in-memory SQL Server data pages, latch enforcement is enhanced with the –T815 trace flag. When a page is latched for modification, the VirtualProtect on the page is set

to PAGE_READWRITE. At all other times the protection is PAGE_READONLY. This can help catch actions such as memory overwrites (scribblers). Starting with...build 8.00.0922, you can dynamically turn on or turn off trace flag -T815 by using the DBCC TRACEON...TRACEOFF. Important Latch enforcement is only valid for non-AWE (Address Windowing Extensions) environments."

IOBasics | CSS_1

818 IO Basics: "...tracks the last 2,048 page write operations. During a successful write I/O completion (proper page ID, bytes transferred successfully, and the proper OS error codes), the DBID, Page ID, and LSN are recorded in a ring buffer. If a failure occurs, error 823 is raised. When an 823 or 605 error is detected, SQL Server looks in the ring buffer for the LSN value that was on the page during the last write. If not correct, extra information is added to the SQL Server error log. The information indicates the type of error along with both the expected and the retrieved LSN."

826433 | 828339 | IOBasics

831 Randal-SQL-SDB407: "Protect unchanged pages in the buffer pool to catch memory corruptions."

Randal-SQL-SDB407

Fixes and Past Relevance

These flags either are old and irrelevant for modern builds, appear only in CTPs, or enable a fix in a CU but are baselined in a later service pack or release.

2509 The only reference I can find is found in Ken Henderson's Guru's Guide to T-SQL, on page 503 (found via books.google.com): "Used in conjunction with DBCC CHECKTABLE to see the total count of ghost records in a table." Maybe the SQL 2000 corollary to 2514 in modern builds?

CHAPTER 18

General Instance-level

CHAPTER 19

SQLOS Scheduling and CPU

CHAPTER 20

SQLOS Memory and Buffer Pool

CHAPTER 21

Disk and Network IO

CHAPTER 22

Background Tasks

CHAPTER 23

Security

CHAPTER 24

Connectivity

CHAPTER 25

HADR / Distributed Technologies

CHAPTER 26

Special Features

CHAPTER 27

Exceptions and Memory Dumping

CHAPTER 28

Miscellaneous Info

CHAPTER 29

Unable To Confirm

CHAPTER 30

Other Stuff

CHAPTER 31

To Do

TODO: incorporate these links:

https://sqlsailor.com/2016/11/07/agdisksector/

http://www.sqlskills.com/blogs/erin/trace-flag-information-in-query-plans/

https://blogs.msdn.microsoft.com/bobsql/2016/11/08/how-it-works-it-just-runs-faster-non-volatile-memory-sql-server-tail-of-log-cach

TODO: catch up on latest hotfixes (for 2014 SP2 CU2), any other releases, TF 2389 additions (see my email), and BWard's 2016 Hekaton slides

TODO: Evaluate combining the following 3 categories since they are all so tightly coupled. Use a special division of flags found only for this super-category.

These categories still need to be pulled in from the Word doc TODO: SQL 2000 Optimization/Query Performance Fixes TODO: SQL 2000 Query Execution TODO: Pre-SQL 2000 Flags TODO: TF 4199 and related TODO: Non-TF4199 Query Performance/Execution fixes TODO: Mis-labeled, unable to find links, Other

Remove these files from my repo, never used OLD: Query Compilation (Info only) and Stats Object-related OLD: Query Compilation Behavioral (Cardinality Estimation only)

Review these links:

https://support.microsoft.com/en-us/kb/3189645

https://support.microsoft.com/en-us/kb/3194716

https://support.microsoft.com/en-us/kb/3195813

https://support.microsoft.com/en-us/kb/3199171

https://support.microsoft.com/en-us/kb/3194717

https://support.microsoft.com/en-us/kb/3182545

https://support.microsoft.com/en-us/kb/3164398

https://support.microsoft.com/en-us/kb/3194718

https://support.microsoft.com/en-us/kb/3194714

https://support.microsoft.com/en-us/kb/3194722

https://sqlperformance.com/2016/11/sql-server-2016/big-deal-sp1

I need to check out the release services link. Also, smoewhere there is a list of TFs and the new USE HINT options they map to, that I need to record.

https://blogs.msdn.microsoft.com/sqlreleaseservices/sql-server-2016-service-pack-1-sp1-released/

https://blogs.msdn.microsoft.com/sql_server_team/query-progress-anytime-anywhere/

https://support.microsoft.com/en-us/kb/3210239?sd=rss&spid=17645

http://sql-sasquatch.blogspot.com/2016/12/retrieve-recent-sql-server-stats-update.html

https://sqlserverscotsman.wordpress.com/2016/12/05/trace-flag-4199-query-optimiser-hotfixes/

https://support.microsoft.com/en-us/kb/3198846

https://blogs.msdn.microsoft.com/sqlcat/2016/12/08/improve-query-performance-on-memory-optimized-tables-with-temporal-using-n

TF 3459 disables parallel redo according to Anthony Nocentino. (I have a screenshot of the Twitter exchange in my "bolex")

https://support.microsoft.com/en-us/help/4010261/sql-server-2016-improvements-in-handling-some-data-types-and-uncommon-

Fantastically-good article (that mentions TF 3656) by Andreas Wolter on diagnosing slow perf with advanced analysis using XEvents, WinDbg, and Wireshark. http://www.sqlservercentral.com/blogs/andreas-wolter-sql-server-bi-blog-english-german/2017/02/02/where-is-that-preemptive-wait-coming-from-database-ownership-and-performance-a-journey-through-sql-server-internals-with-xever

Good demonstration of TF 2453: https://www.brentozar.com/archive/2017/02/using-trace-flag-2453-improve-table-variable-performance/

Review Konstantin Taranov's post on SSC: http://www.sqlservercentral.com/articles/trace-flag/152989/?utm_source=SSC&utm_medium=pubemail

TF 139: https://justdaveinfo.wordpress.com/2017/02/27/sql-server-2016-upgrading-to-compatability-level-130trace-flag-139-and-addi

https://blogs.msdn.microsoft.com/bobsql/2017/05/23/how-it-works-sql-server-deadlock-trace-flag-1222-output/

TF 176 (disables computed column expansion along with some other nuanced behavior). Great Paul White article. https://sqlperformance.com/2017/05/sql-plan/properly-persisted-computed-columns

https://www.sqlskills.com/blogs/erin/query-store-trace-flags/