Automatically Fixing High VLF Count on SQL Server 2012+

FROM: <http://tracyboggiano.com/archive/2017/09/high-vlf-count-fix/>

The Problem

Our databases autogrow because we have thousands of databases and before we know we have databases with high VLF counts in the transaction logs.  If you don’t know why that is bad refer to the resources section of this article.

The Solution

Fix the high VLF count without any manual intervention by the DBA.  This will require a few pieces to the puzzle to work across the various versions of SQL Server.

Step 1

First part of the process if to capture the info from DBCC LOGINFO or if you are ready for 2017 the new dmv sys.dm\_db\_log\_stats into a table you can read later to know how many VLFs exist in your database currently. So we going to create table called VLFInfo and used the procedure VLF\_UpdateInfo to populate that data.  The procedure would be called in step one of a SQL Agent Job to automate the fixing of VLF files during appropriate maintenance windows on your server.  But as you will see in the Step 2 my solutions tries to account for that by not doing them too close to when the file just grew and acquired those new lovely extra VLFs.

VLFInfo Table



|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20 | CREATE TABLE dbo.VLFInfo(  [DBName] [sysname] COLLATE SQL\_Latin1\_General\_CP1\_CI\_AS NOT NULL,  [NumOfVLFs] [int] NOT NULL,  [ActiveVLFs] [int] NOT NULL,  [RecoveryMode] [varchar](7) COLLATE SQL\_Latin1\_General\_CP1\_CI\_AS NOT NULL,  [LogSizeMB] [int] NOT NULL,  [LastUpdateTime] [datetime] NOT NULL CONSTRAINT [DF\_TLogVLFInfo\_LastUpdateTime] DEFAULT (getdate()),  CONSTRAINT [PK\_DBName] PRIMARY KEY CLUSTERED  (  [DBName] ASC  )WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON, FILLFACTOR = 100)  )  GO    CREATE NONCLUSTERED INDEX [IX\_LogSizeMB] ON dbo.[VLFInfo]  (  [LogSizeMB] ASC  )  INCLUDE ([NumOfVLFs]) WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, SORT\_IN\_TEMPDB = OFF, DROP\_EXISTING = OFF, ONLINE = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON, FILLFACTOR = 100)  GO |

VLF\_UpdateInfo stored procedure



|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60  61  62  63  64  65  66  67  68  69  70  71  72  73  74  75  76  77  78  79  80  81  82 | CREATE PROCEDURE dbo.VLF\_UpdateInfo  AS  SET NOCOUNT ON    DECLARE  @DBName SYSNAME,  @vlfcount INT,  @activevlfcount INT,  @dbccquery varchar(1000),  @currentlogsizeMB INT    CREATE TABLE #VLFSummary      (        DBName SYSNAME ,        NumOfVLFs INT ,        ActiveVLfs INT ,        RecoveryMode VARCHAR(99) ,        LogSizeMB INT      )    CREATE TABLE #VLFInfo      (        RecoveryUnitId TINYINT ,        FileId TINYINT ,        FileSize BIGINT ,        StartOffset BIGINT ,        FSeqNo INT ,        [Status] TINYINT ,        Parity TINYINT ,        CreateLSN NUMERIC(25, 0)      )    DECLARE csr CURSOR FAST\_FORWARD READ\_ONLY  FOR  SELECT name  FROM master.sys.databases WHERE database\_id <> 2  AND [state] = 0  AND is\_read\_only = 0    OPEN csr    FETCH NEXT FROM csr INTO @dbname    WHILE (@@fetch\_status <> -1)  BEGIN  SET @dbccquery = REPLACE(REPLACE(  'DBCC loginfo ("{{DatabaseName}}") WITH NO\_INFOMSGS, TABLERESULTS'  ,'"','''')  ,'{{DatabaseName}}', @dbname)    TRUNCATE TABLE #VLFInfo  INSERT INTO #VLFInfo  EXEC (@dbccquery)    SET @vlfcount = @@rowcount    SELECT @activevlfcount = COUNT(\*)  FROM #VLFInfo WHERE [Status] = 2    SELECT @currentlogsizeMB = (size/128)  FROM master.sys.master\_files  WHERE type\_desc = 'log'  AND DB\_NAME(database\_id)=@dbname    INSERT INTO #VLFSummary  VALUES(@dbname, @vlfcount, @activevlfcount, CONVERT(VARCHAR(7),DATABASEPROPERTYEX(@dbname, 'Recovery')), @currentlogsizeMB)    FETCH NEXT FROM csr INTO @dbname  END    CLOSE csr  DEALLOCATE csr    TRUNCATE TABLE dbo.VLFInfo    INSERT INTO dbo.VLFInfo (DBName, NumOfVLFs, ActiveVLFs, RecoveryMode, LogSizeMB)  SELECT DBName, NumOfVLFs, ActiveVLFs, RecoveryMode, LogSizeMB  FROM #VLFSummary    DROP TABLE #VLFSummary  GO |

Step 2

Now that we have that information captured we can execute a stored procedure to fix the high VLF count.  As part of our solution we have an Extended Event session that tracks file growths and shrink activities.  This allows us to run the job all day long and check to see the last time the files have grown.  In our stored procedure VLF\_AutoFix we have a parameter @LookBackTime that is default to 60 minutes, so if the transaction log grew in the last 60 minutes it will not try to fix the high VLFs.

XE\_DatabaseSizeChangeEvents Extended Events



|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17 | CREATE EVENT SESSION XE\_DatabaseSizeChangeEvents      ON SERVER      ADD EVENT sqlserver.database\_file\_size\_change          ( SET collect\_database\_name = ( 1 ))      ADD TARGET package0.ring\_buffer      WITH (   MAX\_MEMORY = 4096KB ,               EVENT\_RETENTION\_MODE = ALLOW\_MULTIPLE\_EVENT\_LOSS ,               MAX\_DISPATCH\_LATENCY = 30 SECONDS ,               MAX\_EVENT\_SIZE = 0KB ,               MEMORY\_PARTITION\_MODE = NONE ,               TRACK\_CAUSALITY = OFF ,               STARTUP\_STATE = ON           );  GO    ALTER EVENT SESSION XE\_DatabaseSizeChangeEvents ON SERVER STATE = start;  GO |

Step 3

Now that we have a way to check to see if the size of the log has changed in the time we specified let’s look at the procedure that we can run to automatically shrink and grow our your log file in the recommended file size chunks. First we are going to create a table to track when we attempt to auto fix our high VLF counts. We will use this table in our procedure to not attempt to auto fix the VLFs for another 24 hours if it was unable to fix it the first time due to active VLF at the end of the transaction log. We don’t the job to constantly be trying to fix those VLFs.

VLFAutoFix table



|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15 | CREATE TABLE dbo.[VLFAutoFix](  [ID] [int] IDENTITY(1,1) NOT NULL,  [DBName] [nvarchar](128) COLLATE SQL\_Latin1\_General\_CP1\_CI\_AS NOT NULL,  [CurrentVLFCount] [int] NOT NULL,  [LogFileSizeMBs] [int] NOT NULL,  [LogDate] [datetime] NOT NULL,  CONSTRAINT [PK\_TLogVLFAutoFix] PRIMARY KEY CLUSTERED  (  [ID] ASC  )WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON, FILLFACTOR = 100)  )  GO    ALTER TABLE dbo.[VLFAutoFix] ADD CONSTRAINT [DF\_VLFAutoFix\_LogDate] DEFAULT (getdate()) FOR [LogDate]  GO |

NOTE: The algorithm for change in 2014 for how VLFs are created so you most likely will not need this procedure for those environments and it doesn’t use a new formula.

Step 4

In the last part we have two procedures.  One serves as a wrapper to call the procedure that actually fix the high VLF count and logs the attempt to our table above.  It also before attempting the fix checks the Extended Event session to see if the log file size changed in the @LookBackTime parameters number of hours and checks the  VLFAutoFix table to see if a fix was attempted the last  @HoursSinceLastFix number of hours. If either those conditions are meet it will not attempt to fix the VLFs and wait until the next time the job runs and check again.  The second procedure actually runs the your backup job and attempts to do the shrinking and growing the file back out.

VLF\_AutoFix stored procedure



|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60  61  62  63  64  65  66  67  68  69  70  71  72  73  74  75  76  77  78  79  80  81  82  83  84  85  86  87  88  89  90  91  92  93  94  95  96  97  98  99  100  101  102  103  104  105  106  107  108  109  110  111  112  113  114  115  116  117  118  119  120  121  122  123  124  125  126  127  128  129  130  131  132  133  134  135  136  137  138  139 | ----------------------------------------------------------------------------------  -- Procedure Name: VLF\_AutoFix  --  -- Desc: Runs VLF\_Fix if Ideal VLFCount is over 100 and VLFCount over IdealCount + @VLFIdealOver  --  --  -- Notes:  Recommendations are to have each VLF be no more than 512 MBs  -- http://www.sqlskills.com/blogs/kimberly/transaction-log-vlfs-too-many-or-too-few/  -- chunks less than 64MB and up to 64MB = 4 VLFs  -- chunks larger than 64MB and up to 1GB = 8 VLFs  -- chunks larger than 1GB = 16 VLFs  -- ideal size for a VLF 512 MBs, 20 to 30 VLfs , 50 high mark  --  https://www.sqlskills.com/blogs/paul/important-change-vlf-creation-algorithm-sql-server-2014/  --  Algorithms update for 2014 and up  --  Is the growth size less than 1/8 the size of the current log size?  --  Yes: create 1 new VLF equal to the growth size  --  No: use the formula above  ----------------------------------------------------------------------------------  CREATE PROCEDURE dbo.VLF\_AutoFix  (  @LookBackTime INT = 60 , -- Number of minutes to look back in the XE to see if things have changed  @VLFIdealOver INT = 20 , -- Number of over the Ideal number of VLFs is OK  @VLFCountMin INT = 100 , -- The minimum of VLFs the log has to have to try to fix it  @VLFIdealSize INT = 512 , -- Ideal size of each VLF  @HoursSinceLastFix INT = 6 , -- Number of hours since the last time it tried to fix it  @MaxIncrementSizeMB INT = 8192, -- Size to increase by, in 2012 and below this 8192 gives is our 512  @LogBackJobName sysname --Name of log backup job on server  )  AS  SET NOCOUNT ON    CREATE TABLE #VLFInfo  (  RecoveryUnitID INT ,  FileID INT ,  FileSize BIGINT ,  StartOffset BIGINT ,  FSeqNo BIGINT ,  [Status] BIGINT ,  Parity BIGINT ,  CreateLSN NUMERIC(38)  );  CREATE TABLE #VLFCountResults  (  DatabaseName SYSNAME ,  VLFCount INT ,  LogFileSize BIGINT  );    CREATE TABLE #Events ( DatabaseName SYSNAME );  CREATE TABLE #LogFileSize ( LogFileSizeMB INT );    DECLARE @DBName SYSNAME ,  @LogFileSize INT ,  @IncrementSizeMB INT ,  @VLFCount INT ,  @SQL NVARCHAR(MAX),  @return\_status int,  @CurrentLogFileSize INT;    DECLARE vlfcursor CURSOR READ\_ONLY  FOR  SELECT  DBName ,  NumOfVLFs ,  LogSizeMB  FROM    dbo.VLFInfo  WHERE   NumOfVLFs > @VLFCountMin  AND NumOfVLFs - ( LogSizeMB / @VLFIdealSize ) >= @VLFIdealOver  ;  OPEN vlfcursor;    FETCH NEXT FROM vlfcursor INTO @DBName, @VLFCount, @LogFileSize ;  WHILE ( @@fetch\_status <> -1 )  BEGIN  IF ( @@fetch\_status <> -2 )  BEGIN  --Query  to see if log files has been grown in the last @LookBackTime Minutes  WITH    Data  AS ( SELECT   CAST(target\_data AS XML) AS TargetData  FROM     sys.dm\_xe\_session\_targets dt  INNER JOIN sys.dm\_xe\_sessions ds ON ds.address = dt.event\_session\_address  WHERE    dt.target\_name = N'ring\_buffer'  AND ds.Name = N'XE\_DatabaseSizeChangeEvents'  )  INSERT INTO #Events  SELECT  XEventData.XEvent.value('(action[@name="database\_name"]/value)[1]', 'SYSNAME') AS DatabaseName  FROM    Data d  CROSS APPLY TargetData.nodes('RingBufferTarget/event[@name=''database\_file\_size\_change'']')  AS XEventData ( XEvent )  WHERE   XEventData.XEvent.value('(@timestamp)[1]', 'datetime2') > CONVERT(DATETIME2, DATEADD(MINUTE, -1 \* @LookBackTime, GETDATE()))  AND XEventData.XEvent.value('(data[@name="file\_type"]/text)[1]', 'NVARCHAR(120)') = N'Log file'  AND XEventData.XEvent.value('(action[@name="database\_name"]/value)[1]', 'SYSNAME') = @DBName;    --If no growths in last @LookBackTime \* -1 minutes then VLF and this process has not been run on this DB in the last @HoursSinceLastFix  IF @@ROWCOUNT = 0 AND NOT EXISTS (SELECT 1 FROM dbo.VLFAutoFix WHERE DBName = @DBName AND LogDate>= DATEADD(HOUR, @HoursSinceLastFix \* -1, GETDATE()))  BEGIN  IF @LogFileSize >= @MaxIncrementSizeMB -- 512 MB limit on VLF size, creates 16 VLFs per growth  SET @IncrementSizeMB = @MaxIncrementSizeMB;  ELSE  SET @IncrementSizeMB = @LogFileSize;  -- Else grow back to original size using size as increment value  --Attempt to shrink and regrow log file  EXEC @return\_status = dbo.VLF\_Fix @DBName = @DBName,  @IncrementSizeMB = @IncrementSizeMB,  @TargetLogSizeMB = @LogFileSize,  @LogBackJobName = @LogBackJobName;    --If previous shrink and regrow was unsuccessful regrow to original size without shrinking                          SELECT  @CurrentLogFileSize = ( size / 128 )                          FROM    master.sys.master\_files                          WHERE   type\_desc = 'log'                                  AND DB\_NAME(database\_id) = @DBName                            IF @LogFileSize > @CurrentLogFileSize  BEGIN                              EXEC dbo.VLF\_Fix                                  @DBName = @DBName ,                                  @IncrementSizeMB = @IncrementSizeMB ,                                  @TargetLogSizeMB = @LogFileSize ,  @LogBackJobName= @LogBackJobName;  END  --Record the Auto Fix info to a table  INSERT INTO dbo.VLFAutoFix (DBName, CurrentVLFCount, LogFileSizeMBs)  VALUES (@DBName, @VLFCount, @LogFileSize);  END    TRUNCATE TABLE #Events;  END  FETCH NEXT FROM vlfcursor INTO @DBName, @VLFCount, @LogFileSize;  END    CLOSE vlfcursor;  DEALLOCATE vlfcursor;    DROP TABLE #VLFInfo;  DROP TABLE #VLFCountResults;  DROP TABLE #Events;  GO |

VLF\_Fix stored procedure



|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60  61  62  63  64  65  66  67  68  69  70  71  72  73  74  75  76  77  78  79  80  81  82  83  84  85  86  87  88  89  90  91  92  93  94  95  96  97  98  99  100  101  102  103  104  105  106  107  108  109  110  111  112  113  114  115  116  117  118  119  120  121  122  123  124  125  126  127  128  129  130  131  132  133  134  135  136  137  138  139  140  141  142  143  144  145  146  147  148  149  150  151  152  153  154  155  156  157  158  159  160  161  162  163  164  165  166  167  168  169  170  171  172  173  174  175  176  177  178  179  180  181  182  183  184  185  186  187  188  189  190  191  192 | CREATE PROCEDURE dbo.VLF\_Fix  (  @DBName sysname,  @StopTimeSecs INT = 600,  @DelayIncrementSecs INT = 1,  @TargetLogSizeMB INT,  @IncrementSizeMB FLOAT = 8192,  @LogBackJobName sysname  )  AS  SET NOCOUNT ON    DECLARE @Delay INT  DECLARE @DelayTime DATETIME  DECLARE @CutOffTime DATETIME  DECLARE @sqlcmd NVARCHAR(MAX)  DECLARE @DBCCQuery VARCHAR(99)  DECLARE @LoopCtr INT  DECLARE @StepMB INT  DECLARE @LogName sysname  DECLARE @CurrentLogFileSizeMB INT  DECLARE @VLFCount INT  DECLARE @DBRecoveryModel CHAR(1)  DECLARE @SQLExceptionMsg VARCHAR(MAX)    CREATE TABLE #VLFInfo      (        RecoveryUnitId TINYINT ,        FileId TINYINT ,        FileSize BIGINT ,        StartOffset BIGINT ,        FSeqNo INT ,        [Status] TINYINT ,        Parity TINYINT ,        CreateLSN NUMERIC(25, 0)      )    -- Set the stop time for the loop  SET @CutOffTime = DATEADD(s,@StopTimeSecs,GETDATE())  SET @Delay = 0  -- Get the recovery model for the database  SET @DBRecoveryModel = ( SELECT ( CASE WHEN d.recovery\_model = 3 THEN 'S' ELSE 'F' END  ) FROM sys.databases AS d WHERE d.name = @DBName )    -- Build SQL command for shrink loop  SET @sqlcmd =  (      SELECT REPLACE('          USE [{{@dbname}}];          CHECKPOINT;          '          ,'{{@dbname}}',@DBName)  ) +  +  REPLACE(REPLACE(  '  SELECT 1  WHILE @@RowCount > 0  BEGIN  SELECT 1  FROM msdb.dbo.sysjobs\_view job  INNER JOIN msdb.dbo.sysjobactivity activity  ON job.job\_id = activity.job\_id  WHERE job.name = "{{JobName}}"  AND start\_execution\_date IS NOT NULL  AND stop\_execution\_date IS NULL  ORDER BY start\_execution\_date DESC  END '  ,'"', '''')  ,'{{JobName}}', @LogBackJobName)  +  REPLACE(REPLACE('  EXEC msdb.dbo.sp\_start\_job "{{JobName}}"'  ,'"', '''')  ,'{{JobName}}', @LogBackJobName)  +  REPLACE(REPLACE(  '  SELECT 1  WHILE @@RowCount > 0  BEGIN  SELECT 1  FROM msdb.dbo.sysjobs\_view job  INNER JOIN msdb.dbo.sysjobactivity activity  ON job.job\_id = activity.job\_id  WHERE job.name = "{{JobName}}"  AND start\_execution\_date IS NOT NULL  AND stop\_execution\_date IS NULL  ORDER BY start\_execution\_date DESC  END'  ,'"', '''')  ,'{{JobName}}', @LogBackJobName)  +  (      SELECT REPLACE('          DBCC SHRINKFILE ({{file\_id}} , 0, TRUNCATEONLY) WITH NO\_INFOMSGS;          CHECKPOINT;  DBCC SHRINKFILE ({{file\_id}} , 0) WITH NO\_INFOMSGS;          '          ,'{{file\_id}}', CONVERT(VARCHAR(99), [file\_id]))      FROM master.sys.master\_files      WHERE type\_desc = 'log'  AND DB\_NAME(database\_id) = @DBName  ) + '  CHECKPOINT;  '    -- Set log name and target size to value of parameter supplied, or existing size if no parameter value supplied  SELECT TOP 1 @LogName = name, @TargetLogSizeMB = ROUND(ISNULL(@TargetLogSizeMB,[size]/128.0),0) FROM master.sys.master\_files WHERE database\_id = DB\_ID(@DBName) AND type = 1 ORDER BY size DESC    -- Get VLF info and store in temporary table  SET @dbccquery = REPLACE(REPLACE(  'DBCC loginfo ("{{DatabaseName}}") WITH NO\_INFOMSGS, TABLERESULTS'  ,'"','''')  ,'{{DatabaseName}}', @dbname)    INSERT INTO #VLFInfo  EXEC (@DBCCQuery)    SELECT @VLFCount = COUNT(\*)  FROM #VLFInfo    SELECT TOP 1 @CurrentLogFileSizeMB = ROUND([size]/128.0,0)  FROM master.sys.master\_files  WHERE database\_id = DB\_ID(@DBName)  AND type = 1  ORDER BY size DESC    -- Run the shrinking loop  WHILE ( (GETDATE()<@CutOffTime) AND ((@CurrentLogFileSizeMB > 100) OR (@VLFCount > 8)) AND (@VLFCount > 2))  BEGIN  -- Run the shrink command only if the most recent log VLF is not active  IF ( (SELECT TOP 1 Status FROM #VLFInfo ORDER BY StartOffset DESC)<>2 )  BEGIN  EXEC sys.sp\_executesql @sqlcmd  END  -- Reset values  TRUNCATE TABLE #VLFInfo    INSERT INTO #VLFInfo  EXEC (@DBCCQuery)    SELECT @VLFCount = COUNT(\*)  FROM #VLFInfo    SELECT TOP 1 @CurrentLogFileSizeMB = ROUND([size]/128.0,0)  FROM master.sys.master\_files  WHERE database\_id = DB\_ID(@DBName)  AND type = 1  ORDER BY size DESC  SET @Delay = @Delay + @DelayIncrementSecs  SET @DelayTime = DATEADD(s,@Delay,GETDATE())  PRINT 'Waiting for ' + CONVERT(VARCHAR(99),@Delay) + ' seconds ...'  PRINT 'Current log file size is ' + CONVERT(VARCHAR(99),@CurrentLogFileSizeMB) + 'MB'  PRINT 'Current VLF count is ' + CONVERT(VARCHAR(99),@VLFCount)    WAITFOR TIME @DelayTime  END    SET @sqlcmd = '-- Target size in MB is ' + ISNULL(CONVERT(VARCHAR(99),@TargetLogSizeMB),'Unknown') + CHAR(13) + CHAR(10)  SET @sqlcmd = @sqlcmd + '-- LogFile name is ' + @LogName + CHAR(13) + CHAR(10)  SET @sqlcmd = @sqlcmd + '-- Ideal increment size is ' + @LogName + CHAR(13) + CHAR(10)    -- Set increment size as close to ideal size as possible (this works better if a target size is supplied that is a multiple of the increment size obviously)  SELECT @StepMB = ROUND(@TargetLogSizeMB / CEILING(@TargetLogSizeMB / @IncrementSizeMB),0)  ,@LoopCtr = CEILING(@TargetLogSizeMB / @IncrementSizeMB)  ,@TargetLogSizeMB = @StepMB    WHILE (@LoopCtr > 0)  BEGIN  SELECT @sqlcmd = @sqlcmd + 'ALTER DATABASE [' + @DBName + '] MODIFY FILE (NAME = N''' + @LogName + ''', SIZE = ' + CONVERT(VARCHAR(9),@TargetLogSizeMB) + 'MB);' + CHAR(13) + CHAR(10)  SELECT @TargetLogSizeMB = @TargetLogSizeMB + @StepMB,@LoopCtr = @LoopCtr - 1  END    IF ( ((GETDATE()<@CutOffTime) AND ((@CurrentLogFileSizeMB <= 100) OR (@VLFCount = 2))))  BEGIN  EXEC sys.sp\_executesql @sqlcmd  END  ELSE  BEGIN  PRINT 'Unable to reduce VLFs sufficiently within the specified time period ...'    PRINT 'Please try again'  SET @SQLExceptionMsg = 'Current log file size is ' + CONVERT(VARCHAR(99),@CurrentLogFileSizeMB) + 'MB'  + 'Current VLF count is ' + CONVERT(VARCHAR(99),@VLFCount)  RAISERROR(@SQLExceptionMsg, 16, 1)  END    DROP TABLE #VLFInfo  GO |

Bringing All Together

Well by now you seen a whole bunch of code and it would be a good time to give you a summary on whole to implement the solution in your environment.  First create your table VLFInfo.  Then create a job to run the first stored procedure VLF\_UpdateInfo to populate that table.  Before you set up the next steps to do any automatic fixes check the VLFInfo across all your servers and see if where you can tests this at.

Next you need to create the Extended Event XE\_DatabaseSizeChangeEvents session and let it start collecting data for you.  Then we can create the tracking table VLFAutoFix so we will know when it attempts to fix high VLF counts.  Then create stored procedures VLF\_AutoFix and VLF\_Fix.  Then as part of the job you created setup step two to execute procedure VLF\_AutoFix.

Now create a SQL Agent job.  Step one will call procedure VLF\_UpdateInfo.  Step two will call VLF\_AutoFix with the parameter @LogBackJobName parameter specifying your log back job so it can help mark the VLFs a the end the log file not active..

Currently this system runs hourly across 400 servers over thousands of databases with 30K+ transactions per second without causing any issues because it does take the precautions not to run it if the file has grown in the last hour and to not attempt to fix it within 6 hours of the previous attempt.

Resources

[Transaction Log VLFs – too many or too few?](https://www.sqlskills.com/blogs/kimberly/transaction-log-vlfs-too-many-or-too-few/)

[8 Steps to better Transaction Log throughput](https://www.sqlskills.com/blogs/kimberly/8-steps-to-better-transaction-log-throughput/)

[Performance impact of lots of VLFs in the transaction log](https://www.sqlskills.com/blogs/paul/performance-impact-of-lots-of-vlfs-in-the-transaction-log/)

11 thoughts on “Automatically Fixing High VLF Count on SQL Server 2012+”

1. Pingback: [Automatically Fix Those VLFs – Curated SQL](https://curatedsql.com/2017/09/20/automatically-fix-those-vlfs/)
2. http://2.gravatar.com/avatar/b6d6a9f262fb828ec84933f4902a8fc1?s=60&d=mm&r=gOZ says:

[September 25, 2017 at 11:01](http://tracyboggiano.com/archive/2017/09/high-vlf-count-fix/#comment-506)

Nice article Tracy, I might roll this one out too! Tiny typo in title, replace “Sever” with “Server” ^\_^

[Reply](http://tracyboggiano.com/archive/2017/09/high-vlf-count-fix/?replytocom=506#respond)

* 1. http://1.gravatar.com/avatar/a63f42cb9666ae6b2c49b48f505c860f?s=60&d=mm&r=g[Tracy Boggiano](http://tracyboggiano.com/) says:

[September 25, 2017 at 11:38](http://tracyboggiano.com/archive/2017/09/high-vlf-count-fix/#comment-507)

Thanks I always have at least one typo.

[Reply](http://tracyboggiano.com/archive/2017/09/high-vlf-count-fix/?replytocom=507#respond)

1. http://1.gravatar.com/avatar/12831ce1edf4876220b4d718a9913454?s=60&d=mm&r=gPS says:

[September 25, 2017 at 13:32](http://tracyboggiano.com/archive/2017/09/high-vlf-count-fix/#comment-508)

Tracy,

Are you sure this works on 2008 SQL Srvr? Particularly the XE session in step 2. I just tried to run Step 2 on a SQL Server 2008 R2 database, and it fails with the message “The event name, “sqlserver.database\_file\_size\_change”, is invalid, or the object could not be found”.

Perhaps SS 2012 or greater… I haven’t tried this yet on a 2012 instance.

[Reply](http://tracyboggiano.com/archive/2017/09/high-vlf-count-fix/?replytocom=508#respond)

* 1. http://1.gravatar.com/avatar/a63f42cb9666ae6b2c49b48f505c860f?s=60&d=mm&r=g[Tracy Boggiano](http://tracyboggiano.com/) says:

[September 25, 2017 at 16:51](http://tracyboggiano.com/archive/2017/09/high-vlf-count-fix/#comment-512)

Your right I think it is 2012 and up. I forgot when I actually wrote this but looked when the extended event was created and confirmed it is in 2012.

[Reply](http://tracyboggiano.com/archive/2017/09/high-vlf-count-fix/?replytocom=512#respond)

1. http://0.gravatar.com/avatar/61912ea67ba9f882256dfe5bdfd76c0b?s=60&d=mm&r=gSteve says:

[September 25, 2017 at 13:48](http://tracyboggiano.com/archive/2017/09/high-vlf-count-fix/#comment-509)

I implemented something similar in my shop, only it warns of VLF issues instead of fixing them. The option is there to auto fix, but it is turned off. We have a lot of log reader sensitive activity – cdc, log shipping, etc. and shrinking a log at the wrong time can be problematic.

[Reply](http://tracyboggiano.com/archive/2017/09/high-vlf-count-fix/?replytocom=509#respond)

* 1. http://1.gravatar.com/avatar/a63f42cb9666ae6b2c49b48f505c860f?s=60&d=mm&r=g[Tracy Boggiano](http://tracyboggiano.com/) says:

[September 25, 2017 at 16:53](http://tracyboggiano.com/archive/2017/09/high-vlf-count-fix/#comment-513)

We started off that way with warnings but found with the extra logic of not doing if the log grew recently we didn’t have any issues with it. We have AGs that we run it with no issues and replication.

[Reply](http://tracyboggiano.com/archive/2017/09/high-vlf-count-fix/?replytocom=513#respond)

1. http://1.gravatar.com/avatar/7fcd86a197cd12ccf634dd32878442a4?s=60&d=mm&r=gJames says:

[September 25, 2017 at 13:50](http://tracyboggiano.com/archive/2017/09/high-vlf-count-fix/#comment-510)

For the Agent job, how frequent is it ran?

[Reply](http://tracyboggiano.com/archive/2017/09/high-vlf-count-fix/?replytocom=510#respond)

* 1. http://1.gravatar.com/avatar/a63f42cb9666ae6b2c49b48f505c860f?s=60&d=mm&r=g[Tracy Boggiano](http://tracyboggiano.com/) says:

[September 25, 2017 at 16:54](http://tracyboggiano.com/archive/2017/09/high-vlf-count-fix/#comment-514)

We run ours every hour as it only fix it when it finds one that is over the limit that hasn’t grown the last hour.

[Reply](http://tracyboggiano.com/archive/2017/09/high-vlf-count-fix/?replytocom=514#respond)

1. http://2.gravatar.com/avatar/be9d62ff4d79a493720b0ca4d8381036?s=60&d=mm&r=gTom says:

[September 25, 2017 at 14:49](http://tracyboggiano.com/archive/2017/09/high-vlf-count-fix/#comment-511)

Tracy,

Nice job! Could you explain more about why “not to run it if the file has grown in the last hour and to not attempt to fix it within 6 hours of the previous attempt.”?

Thanks,  
Tom

[Reply](http://tracyboggiano.com/archive/2017/09/high-vlf-count-fix/?replytocom=511#respond)

* 1. http://1.gravatar.com/avatar/a63f42cb9666ae6b2c49b48f505c860f?s=60&d=mm&r=g[Tracy Boggiano](http://tracyboggiano.com/) says:

[September 25, 2017 at 16:57](http://tracyboggiano.com/archive/2017/09/high-vlf-count-fix/#comment-515)

Yes, we don’t run if it has grown in the last hour just in case the activity that caused it to grow is continuing to happen so we want the activity to settle down. We wait the 6 hours because we don’t want to constantly trying to run shrink commands every hour while we wait for the active VLF to be wrap around to the beginning of the file. It’s just a safety measure to not have too much maintenance activity occurring on the system.