WSUS Cleanup Wizard Fails and stops WSUS Service

Problem

I've been having issues with a WSUS Server recently, when I go to clean-up the server and reclaim some space from unused/superseded updates. ~10/13/2019

The symptoms are as follows:

Running the Cleanup Wizard hangs at Deleting Unused Updates, and then the WSUS Console fails, and the UI asks you to "Reset Node". Resetting the node restarts the service, and all is well until you try to run the Cleanup Wizard again and try to Delete Updates but will loop on:

- Hang UI
- WSUS Service stops
- UI request you to reset node.

Analysis

When you start digging into the WSUS logs:

- **C:\Program Files\UpdateServices\LogFiles\SoftwareDistribution.log**

…You find out there was a timeout exception when WSUS ran the procedure:

- **spGetObsoleteUpdatesToCleanup**

2019-10-14 06:06:17.111 UTC Warning w3wp.349 SoapExceptionProcessor.SerializeAndThrow Discarding stack trace for user DOMAIN\Administrator, IP Address ::1, exception System.Data.SqlClient.SqlException (0x80131904): Execution Timeout Expired. The timeout period elapsed prior to completion of the operation or the server is not responding. ---> System.ComponentModel.Win32Exception (0x80004005): The wait operation timed out

at Microsoft.UpdateServices.DatabaseAccess.DBConnection.DrainObsoleteConnections(SqlException e)

at Microsoft.UpdateServices.DatabaseAccess.DBConnection.ExecuteReader()

at Microsoft.UpdateServices.Internal.SingleResultSetSPHandler.ExecuteStoredProcedure(DBConnection connection)

at Microsoft.UpdateServices.Internal.GenericDataAccess.ExecuteSP(String spName, DBParameterCollection args, IExecuteSPHandler handler, Int32 queryTimeoutInSeconds)

at Microsoft.UpdateServices.Internal.GenericDataAccess.ExecuteSP(String spName, DBParameterCollection args, IExecuteSPHandler handler)

at Microsoft.UpdateServices.Internal.DatabaseAccess.CommonDataAccess.ExecuteSPSingleResultSet(String spName, DBParameterCollection args, Type resultType, Int32 queryTimeoutInSeconds)

at Microsoft.UpdateServices.Internal.DatabaseAccess.CommonDataAccess.ExecuteSPSingleResultSet(String spName, DBParameterCollection args, Type resultType)

at Microsoft.UpdateServices.Internal.DatabaseAccess.AdminDataAccess.ExecuteSPGetObsoleteUpdatesToCleanup()

at Microsoft.UpdateServices.Internal.ApiRemoting.ExecuteSPGetObsoleteUpdatesToCleanup()

When you go to SQL Server Management Studio, and run that procedure manually, it runs well beyond 2 minutes (ASP timeout), and as such the query fails because it takes longer than expected. In fact, I've let it run for 25 minutes or more and it didn't return the expected result set and had to cancel that query.

In examining the OOTB code, we find that it is basically, looking at a few tables and returning a single column "LocalUpdateID" after pruning the results based on a few where clauses as follows:

```
ALTER PROCEDURE [dbo].[spGetObsoleteUpdatesToCleanup]
AS
SET NOCOUNT ON
DECLARE @minimumDeadDeploymentTime DATETIME
DECLARE @revisionDeletionTimeThreshold INT
SELECT @revisionDeletionTimeThreshold=RevisionDeletionTimeThreshold FROM
dbo.tbConfigurationC
IF @@ERROR <> 0
BEGIN
    RAISERROR('spGetObsoleteUpdatesToCleanup: failed to get RevisionDeletionTimeThreshold
from dbo.tbConfigurationC', 16, -1)
    RETURN (1)
END
SET @minimumDeadDeploymentTime = DATEADD(day, 0 - @revisionDeletionTimeThreshold,
getutcdate())
SELECT DISTINCT u.LocalUpdateID FROM  dbo.tbUpdate u
    INNER JOIN dbo.tbRevision r ON r.LocalUpdateID = u.LocalUpdateID
    INNER JOIN dbo.tbProperty p ON p.RevisionID = r.RevisionID
WHERE
     p.PublicationState = 1
     AND (p.ExplicitlyDeployable = 1 OR p.UpdateType IN ('Category', 'Detectoid'))
     AND p.ReceivedFromCreatorService <= @minimumDeadDeploymentTime
     AND NOT EXISTS (SELECT * FROM dbo.tbBundleDependency bd
                     INNER JOIN dbo.tbRevision r1 ON bd.BundledRevisionID = r1.RevisionID
                     WHERE r1.LocalUpdateID = u.LocalUpdateID)
     AND NOT EXISTS (SELECT * FROM dbo.tbPrerequisiteDependency pd
                     INNER JOIN dbo.tbRevision r2 ON pd.PrerequisiteRevisionID =
r2.RevisionID
                     WHERE r2.LocalUpdateID = u.LocalUpdateID)
     AND NOT EXISTS (SELECT * FROM dbo.tbDeployment d
                     INNER JOIN dbo.tbRevision r3 ON d.RevisionID = r3.RevisionID
                     WHERE r3.LocalUpdateID = u.LocalUpdateID
                         AND d.TargetGroupTypeID = 0
                         AND d.ActionID IN (0, 1, 3))
     AND NOT EXISTS (SELECT * FROM dbo.tbDeadDeployment dd
                     INNER JOIN dbo.tbRevision r4 ON dd.RevisionID = r4.RevisionID
                     WHERE r4.LocalUpdateID = u.LocalUpdateID
                         AND dd.TargetGroupTypeID = 0
                         AND dd.ActionID IN (0, 1, 3)
                         AND dd.TimeOfDeath > @minimumDeadDeploymentTime)
ORDER BY u.LocalUpdateID DESC
RETURN (0)

GO
```

If we grab the first part of this query (below), it seems to return the results fine and relatively quickly returns the results. **However, that is also dependant on what the WSUS Server is doing at the time (see notes below)…**

```
SET NOCOUNT ON
DECLARE @minimumDeadDeploymentTime DATETIME
DECLARE @revisionDeletionTimeThreshold INT
SELECT @revisionDeletionTimeThreshold=RevisionDeletionTimeThreshold FROM
dbo.tbConfigurationC
IF @@ERROR <> 0
BEGIN
    RAISERROR('spGetObsoleteUpdatesToCleanup: failed to get RevisionDeletionTimeThreshold
from dbo.tbConfigurationC', 16, -1)
    RETURN (1)
END
SET @minimumDeadDeploymentTime = DATEADD(day, 0 - @revisionDeletionTimeThreshold,
getutcdate())
SELECT DISTINCT u.LocalUpdateID FROM  dbo.tbUpdate u
    INNER JOIN dbo.tbRevision r ON r.LocalUpdateID = u.LocalUpdateID
    INNER JOIN dbo.tbProperty p ON p.RevisionID = r.RevisionID
WHERE
     p.PublicationState = 1
     AND (p.ExplicitlyDeployable = 1 OR p.UpdateType IN ('Category', 'Detectoid'))
     AND p.ReceivedFromCreatorService <= @minimumDeadDeploymentTime
     AND NOT EXISTS (SELECT * FROM dbo.tbBundleDependency bd
                     INNER JOIN dbo.tbRevision r1 ON bd.BundledRevisionID = r1.RevisionID
                     WHERE r1.LocalUpdateID = u.LocalUpdateID)
```

I found that this query can take 25+ minutes to run:

However, at startup, WSUS queries the tbUpdates table for all the UpdateIDs, and can cause issues with this query, which was some of the intermittent behavior I was seeing while working on this problem.

After starting the WSUS Service, This log entry should indicate that the server has successfully performed it's lookup, and it's now safe to query that table again.

2019-10-15 05:12:38.192 UTC Info w3wp.36 DependencyCache.RefreshThreadStart read 396699 UpdateIDs from DB in 547 ms

Once this log entry shows up in the SoftwareDistribution.log, then you should be able to use the WSUS Cleanup Wizard to run the necessary cleanups, but I would almost argue that we need to add something to the procedure to avoid that type of race condition:

i.e. `SET TRANSACTION ISOLATION LEVEL READ UNCOMMITTED`

Once you can query the tbUpdates table for the LocalUpdatesID, it might take HOURS/Days to delete the various updates depending on exactly how many updates it's trying to cleanup. In my case, it was trying to delete 2700+ updates. From that perspective, it might be easier to monitor this process using this SQL code. The following code is slightly different from `spGetObsoleteUpdatesToCleanup` with logic

to incorporate spDeleteUpdate from a cursor to delete the obsolete Updates and show you what updates it is deleting (as denoted by the LocalUpdateID). In my troubleshooting, many of the updates were related to Windows 10 Cumulative updates, or to Windows Defender Security Intelligence updates (definition updates).

In diagnosing this problem, it is apparent that the sheer number of Revisions to the Windows Defender Definition files are in part the cause of the problem with the Cleanup Server Wizard crashing (when having WSUS supply Windows Defender Definition updates).

NOTE: An updated version of this code that allows you to Display or Display/Delete the Obsolete Updates is available here: WSUSGetObsoleteUpdatesAndDeleteThem.sql

```
SET TRANSACTION ISOLATION LEVEL READ UNCOMMITTED

IF OBJECT_ID('tempdb..#localupdateids') IS NOT NULL
    DROP TABLE #localupdateids
--ENDIF

CREATE TABLE #localupdateids (LocalUpdateID INT, UpdateTitle varchar(200))

SET NOCOUNT ON

DECLARE @minimumDeadDeploymentTime DATETIME
DECLARE @revisionDeletionTimeThreshold INT
SELECT @revisionDeletionTimeThreshold=RevisionDeletionTimeThreshold FROM
dbo.tbConfigurationC
IF @@ERROR <> 0
       BEGIN
              RAISERROR('spGetObsoleteUpdatesToCleanup: failed to get
RevisionDeletionTimeThreshold from dbo.tbConfigurationC', 16, -1)
       END
ELSE
    BEGIN
              SET @minimumDeadDeploymentTime = DATEADD(day, 0 -
@revisionDeletionTimeThreshold, getutcdate())

              INSERT INTO #localupdateids

              SELECT DISTINCT u.LocalUpdateID,PCLP.Title as UpdateTitle FROM
dbo.tbUpdate u
                     INNER JOIN dbo.tbRevision r ON r.LocalUpdateID = u.LocalUpdateID --
attach revisions for further joins
                     INNER JOIN dbo.tbProperty p ON p.RevisionID = r.RevisionID -- attach
the properties for further joins
                     INNER JOIN dbo.tbPreComputedLocalizedProperty PCLP  ON
PCLP.UpdateID=U.UpdateID  -- attach PCLP so we can get the Update Title
                     INNER JOIN dbo.tbLanguage L on L.ShortLanguage = PCLP.ShortLanguage
and L.Enabled = 1 -- attach Languages so we can find out what language(s) is/are
currently subscribed
                     INNER JOIN dbo.tbLanguageInSubscription LIS on LIS.LanguageID =
L.LanguageID  -- filter out language names that are not subscribed (so we don't end up
with update names in an unexpected language)
                         --note multiple subscribed languages will mess with the logic
here because it will try to delete the same update twice (once for each language)
              WHERE
```

```
                    p.PublicationState = 1
                    AND (p.ExplicitlyDeployable = 1 OR p.UpdateType IN ('Category',
'Detectoid'))
                    AND p.ReceivedFromCreatorService <= @minimumDeadDeploymentTime

            --reworked the AND NOT EXISTS from spGetObsoleteUpdatesToCleanup into these
deletes from my temp table as a workaround for the enormous amount of time that the
original procedure takes to find obsolete updates

            --remove updates from the list that are dependencies
            DELETE from #localupdateids
              WHERE LocalUpdateID IN (SELECT LocalUpdateID FROM dbo.tbBundleDependency
bd INNER JOIN dbo.tbRevision r1 ON bd.BundledRevisionID = r1.RevisionID)

            --remove updates from the list that are prerequisites
            DELETE from #localupdateids
              WHERE LocalUpdateID IN (SELECT LocalUpdateID FROM
dbo.tbPrerequisiteDependency pd INNER JOIN dbo.tbRevision r2 ON pd.PrerequisiteRevisionID
= r2.RevisionID)

            --remove updates from the list that are currently being deployed
            DELETE from #localupdateids
              WHERE LocalUpdateID IN (SELECT LocalUpdateID FROM dbo.tbDeployment d
INNER JOIN dbo.tbRevision r3 ON d.RevisionID = r3.RevisionID WHERE d.TargetGroupTypeID =
0 AND d.ActionID IN (0, 1, 3))

            --remove updates from the list that are "dead deployments"?
            DELETE from #localupdateids
              WHERE LocalUpdateID IN (SELECT LocalUpdateID FROM dbo.tbDeadDeployment dd
INNER JOIN dbo.tbRevision r4 ON dd.RevisionID = r4.RevisionID WHERE dd.TargetGroupTypeID
= 0 AND dd.ActionID IN (0, 1, 3) AND dd.TimeOfDeath > @minimumDeadDeploymentTime)

            select * from #localupdateids
            DECLARE @max_title_length INT
            SELECT @max_title_length = max(len(UpdateTitle)) from #localupdateids
            select @max_title_length

            --now run through the list of updates and delete each update (and ALL THE
REVISIONS  -- code in spDeleteUpdate handles that)
            DECLARE @msg nvarchar(1000)
            DECLARE @update_title varchar(400)
            DECLARE @update_id INT
            DECLARE @curitem INT, @totaltodelete INT
            SET @totaltodelete = (SELECT COUNT(*) FROM #localupdateids)
            SELECT @curitem=1

            SET @msg = 'Total Updates To Delete = ' + cast(@totaltodelete as
varchar(5)) + ' (processing in reverse order)' RAISERROR(@msg,0,1) WITH NOWAIT

            DECLARE WC Cursor FOR SELECT LocalUpdateID, UpdateTitle FROM
#localupdateids order by LocalUpdateID DESC

            OPEN WC
            FETCH NEXT FROM WC INTO @update_id,@update_title WHILE (@@FETCH_STATUS > -
1)
              BEGIN
                  SET @msg = cast(getdate() as varchar(30)) + ' - ' + replicate(' ',5
- len(cast(@curitem as varchar(5)))) + cast(@curitem as varchar(5)) + '/' +
```

```
cast(@totaltodelete as varchar(5)) + ': Deleting ''' + @update_title + '''' + replicate('
', @max_title_length - len(@update_title)) + ' -> ' + CONVERT(varchar(10), @update_id)
                    RAISERROR(@msg,0,1) WITH NOWAIT

                    EXEC spDeleteUpdate @localUpdateID = @update_id
                    --RAISERROR('   waiting 5 minutes',0,1) WITH NOWAIT
                    --WAITFOR DELAY '00:05:00'
                    SET @curitem = @curitem +1
                    FETCH NEXT FROM WC INTO @update_id, @update_title
                END
            CLOSE WC

            DEALLOCATE WC

            DROP TABLE #localupdateids
        END
--ENDIF
```

While running this SQL code above, you should see log entries in the SoftwareDistribution.log similar to below every time an Update is deleted as denoted in the Query Pane on the “Messages” tab of SQL Server Management Studio… If not, that can indicate an issue with WSUS, and indicate you should restart WSUS, IIS, and/or SQL Server and then wait for the log entry indicating that WSUS successfully looked up all the UpdateIDs in nnn miliseconds (as shown above) before trying to delete Obsolete Updates again:

2019-10-15 05:39:50.495 UTC Info WsusService.8 SusEventDispatcher.TriggerEventTriggerEvent called for NotificationEventName: DeploymentChange, EventInfo: DeploymentChange

2019-10-15 05:39:50.495 UTC Info w3wp.388 ThreadEntry ThreadHelper.ThreadStart

2019-10-15 05:39:50.511 UTC Info w3wp.388 SusEventDispatcher.DispatchManagerWorkerThreadProc DispatchManager Worker Thread Processing NotificationEvent: DeploymentChange

2019-10-15 05:39:50.511 UTC Info w3wp.388 DeploymentChangeNotification.InternalEventHandler deployment change event received

2019-10-15 05:39:50.511 UTC Info w3wp.388 RevisionIdCacheChangeNotificationDispatcher.InternalEventHandler Get event DeploymentChange from dispatchmanager

2019-10-15 05:39:50.511 UTC Info w3wp.10 SusEventDispatcher.TriggerEventTriggerEvent called for NotificationEventName: DeploymentChange, EventInfo: DeploymentChange

2019-10-15 05:39:50.511 UTC Info w3wp.391 ThreadEntry ThreadHelper.ThreadStart

2019-10-15 05:39:50.511 UTC Info w3wp.391 SusEventDispatcher.DispatchManagerWorkerThreadProc DispatchManager Worker Thread Processing NotificationEvent: DeploymentChange

2019-10-15 05:39:50.511 UTC Info w3wp.391 ChangeNotificationDispatcher.InternalEventHandler Get event DeploymentChange from dispatchmanager

Keep in mind that Updates like Windows Defender (that update daily, or even hourly) pretty much cause WSUS to have issues with the WSUS Server Cleanup tool because of their sheer number of updates and the fact that any particular update has close to 20 files attached. As well, the number of revisions to a single update undoubtedly cause issues to the SQL Code when looking up dependency/superseding information. The "SLIM" and "DELTA" files for 2 Windows Defender definition updates are shown here:

Record count: 38

Update Platform	Update Title	File Name	GB Downloaded	Total GB For Download	Percent Complete
Windows Defender	Security Intelligence Update for Windows Defender Antivirus - KB2267602 (Version 1.303.1744.0)	AM_Delta_Patch_1.303.1697.0.exe	0.00GB	0.00GB	100%
Windows Defender	Security Intelligence Update for Windows Defender Antivirus - KB2267602 (Version 1.303.1744.0)	AM_Delta_Patch_1.303.1727.0.exe	0.00GB	0.00GB	100%
Windows Defender	Security Intelligence Update for Windows Defender Antivirus - KB2267602 (Version 1.303.1744.0)	AM_Delta_Patch_1.303.1735.0.exe	0.00GB	0.00GB	100%
Windows Defender	Update for Windows Defender Antivirus antimalware platform - KB4052623 (Version 4.18.1910.3)	updateplatform.exe	0.00GB	0.00GB	100%
Windows Defender	Security Intelligence Update for Windows Defender Antivirus - KB2267602 (Version 1.303.1744.0)	AM_Delta.exe	0.00GB	0.02GB	0%
Windows Defender	Security Intelligence Update for Windows Defender Antivirus - KB2267602 (Version 1.303.1744.0)	AM_Delta_Patch_1.303.1681.0.exe	0.00GB	0.00GB	0%
Windows Defender	Security Intelligence Update for Windows Defender Antivirus - KB2267602 (Version 1.303.1744.0)	AM_Delta_Patch_1.303.1684.0.exe	0.00GB	0.00GB	0%
Windows Defender	Security Intelligence Update for Windows Defender Antivirus - KB2267602 (Version 1.303.1744.0)	AM_Delta_Patch_1.303.1688.0.exe	0.00GB	0.00GB	0%
Windows Defender	Security Intelligence Update for Windows Defender Antivirus - KB2267602 (Version 1.303.1744.0)	AM_Delta_Patch_1.303.1692.0.exe	0.00GB	0.00GB	0%
Windows Defender	Security Intelligence Update for Windows Defender Antivirus - KB2267602 (Version 1.303.1744.0)	AM_Delta_Patch_1.303.1697.0.exe	0.00GB	0.00GB	0%
Windows Defender	Security Intelligence Update for Windows Defender Antivirus - KB2267602 (Version 1.303.1744.0)	AM_Delta_Patch_1.303.1700.0.exe	0.00GB	0.00GB	0%
Windows Defender	Security Intelligence Update for Windows Defender Antivirus - KB2267602 (Version 1.303.1744.0)	AM_Delta_Patch_1.303.1704.0.exe	0.00GB	0.00GB	0%
Windows Defender	Security Intelligence Update for Windows Defender Antivirus - KB2267602 (Version 1.303.1744.0)	AM_Delta_Patch_1.303.1708.0.exe	0.00GB	0.00GB	0%
Windows Defender	Security Intelligence Update for Windows Defender Antivirus - KB2267602 (Version 1.303.1744.0)	AM_Delta_Patch_1.303.1713.0.exe	0.00GB	0.00GB	0%
Windows Defender	Security Intelligence Update for Windows Defender Antivirus - KB2267602 (Version 1.303.1744.0)	AM_Delta_Patch_1.303.1716.0.exe	0.00GB	0.00GB	0%
Windows Defender	Security Intelligence Update for Windows Defender Antivirus - KB2267602 (Version 1.303.1744.0)	AM_Delta_Patch_1.303.1721.0.exe	0.00GB	0.00GB	0%
Windows Defender	Security Intelligence Update for Windows Defender Antivirus - KB2267602 (Version 1.303.1744.0)	AM_Delta_Patch_1.303.1724.0.exe	0.00GB	0.00GB	0%
Windows Defender	Security Intelligence Update for Windows Defender Antivirus - KB2267602 (Version 1.303.1744.0)	AM_Delta_Patch_1.303.1727.0.exe	0.00GB	0.00GB	0%
Windows Defender	Security Intelligence Update for Windows Defender Antivirus - KB2267602 (Version 1.303.1744.0)	AM_Delta_Patch_1.303.1731.0.exe	0.00GB	0.00GB	0%
Windows Defender	Security Intelligence Update for Windows Defender Antivirus - KB2267602 (Version 1.303.1744.0)	AM_Delta_Patch_1.303.1735.0.exe	0.00GB	0.00GB	0%
Windows Defender	Security Intelligence Update for Windows Defender Antivirus - KB2267602 (Version 1.303.1744.0)	AM_Delta_Patch_1.303.1742.0.exe	0.00GB	0.00GB	0%
Windows Defender	Security Intelligence Update for Windows Defender Antivirus - KB2267602 (Version 1.303.1744.0)	AM_Slim_Delta.exe	0.00GB	0.01GB	0%
Windows Defender	Security Intelligence Update for Windows Defender Antivirus - KB2267602 (Version 1.303.1744.0)	AM_Slim_Delta_Patch_1.303.1681.0.exe	0.00GB	0.00GB	0%
Windows Defender	Security Intelligence Update for Windows Defender Antivirus - KB2267602 (Version 1.303.1744.0)	AM_Slim_Delta_Patch_1.303.1684.0.exe	0.00GB	0.00GB	0%
Windows Defender	Security Intelligence Update for Windows Defender Antivirus - KB2267602 (Version 1.303.1744.0)	AM_Slim_Delta_Patch_1.303.1688.0.exe	0.00GB	0.00GB	0%
Windows Defender	Security Intelligence Update for Windows Defender Antivirus - KB2267602 (Version 1.303.1744.0)	AM_Slim_Delta_Patch_1.303.1692.0.exe	0.00GB	0.00GB	0%
Windows Defender	Security Intelligence Update for Windows Defender Antivirus - KB2267602 (Version 1.303.1744.0)	AM_Slim_Delta_Patch_1.303.1697.0.exe	0.00GB	0.00GB	0%
Windows Defender	Security Intelligence Update for Windows Defender Antivirus - KB2267602 (Version 1.303.1744.0)	AM_Slim_Delta_Patch_1.303.1700.0.exe	0.00GB	0.00GB	0%
Windows Defender	Security Intelligence Update for Windows Defender Antivirus - KB2267602 (Version 1.303.1744.0)	AM_Slim_Delta_Patch_1.303.1704.0.exe	0.00GB	0.00GB	0%
Windows Defender	Security Intelligence Update for Windows Defender Antivirus - KB2267602 (Version 1.303.1744.0)	AM_Slim_Delta_Patch_1.303.1708.0.exe	0.00GB	0.00GB	0%
Windows Defender	Security Intelligence Update for Windows Defender Antivirus - KB2267602 (Version 1.303.1744.0)	AM_Slim_Delta_Patch_1.303.1713.0.exe	0.00GB	0.00GB	0%
Windows Defender	Security Intelligence Update for Windows Defender Antivirus - KB2267602 (Version 1.303.1744.0)	AM_Slim_Delta_Patch_1.303.1716.0.exe	0.00GB	0.00GB	0%
Windows Defender	Security Intelligence Update for Windows Defender Antivirus - KB2267602 (Version 1.303.1744.0)	AM_Slim_Delta_Patch_1.303.1721.0.exe	0.00GB	0.00GB	0%
Windows Defender	Security Intelligence Update for Windows Defender Antivirus - KB2267602 (Version 1.303.1744.0)	AM_Slim_Delta_Patch_1.303.1724.0.exe	0.00GB	0.00GB	0%
Windows Defender	Security Intelligence Update for Windows Defender Antivirus - KB2267602 (Version 1.303.1744.0)	AM_Slim_Delta_Patch_1.303.1727.0.exe	0.00GB	0.00GB	0%
Windows Defender	Security Intelligence Update for Windows Defender Antivirus - KB2267602 (Version 1.303.1744.0)	AM_Slim_Delta_Patch_1.303.1731.0.exe	0.00GB	0.00GB	0%
Windows Defender	Security Intelligence Update for Windows Defender Antivirus - KB2267602 (Version 1.303.1744.0)	AM_Slim_Delta_Patch_1.303.1735.0.exe	0.00GB	0.00GB	0%
Windows Defender	Security Intelligence Update for Windows Defender Antivirus - KB2267602 (Version 1.303.1744.0)	AM_Slim_Delta_Patch_1.303.1742.0.exe	0.00GB	0.00GB	0%

NOTE: This is a screenshot from a sample web report for something I created to view currently downloading patch files to WSUS and shows the number of patch files related to 2 patches that were auto-approved by some automation I wrote.

Regular Maintenance

It's also good to note that the following script should be run with regularity to ensure that the WSUS Database is as optimized as it can be from a SQL index and statistics perspective.

This particular example was derived from the Microsoft SQL Server help files (with some minor changes/updates)

```
/*Perform a 'USE <database name>' to select the database in which to run the script.*/
USE SUSDB

sp_updatestats 'resample'
```

```
GO
-- Declare variables
SET NOCOUNT ON
DECLARE @tablename VARCHAR (128)
DECLARE @execstr   VARCHAR (255)
DECLARE @objectid  INT
DECLARE @indexid   INT
DECLARE @frag      DECIMAL
DECLARE @maxfrag   DECIMAL

-- Decide on the maximum fragmentation to allow
SELECT @maxfrag = 1.0

-- Declare cursor
DECLARE tables CURSOR FOR
   SELECT TABLE_NAME
   FROM INFORMATION_SCHEMA.TABLES
   WHERE TABLE_TYPE = 'BASE TABLE'

-- Create the table
CREATE TABLE #fraglist (
   ObjectName CHAR (255),
   ObjectId INT,
   IndexName CHAR (255),
   IndexId INT,
   Lvl INT,
   CountPages INT,
   CountRows INT,
   MinRecSize INT,
   MaxRecSize INT,
   AvgRecSize INT,
   ForRecCount INT,
   Extents INT,
   ExtentSwitches INT,
   AvgFreeBytes INT,
   AvgPageDensity INT,
   ScanDensity DECIMAL,
   BestCount INT,
   ActualCount INT,
   LogicalFrag DECIMAL,
   ExtentFrag DECIMAL)

-- Open the cursor
OPEN tables

-- Loop through all the tables in the database
FETCH NEXT
   FROM tables
   INTO @tablename

WHILE @@FETCH_STATUS = 0
BEGIN
-- Do the showcontig of all indexes of the table
   INSERT INTO #fraglist
   EXEC ('DBCC SHOWCONTIG (''' + @tablename + ''')
      WITH FAST, TABLERESULTS, ALL_INDEXES, NO_INFOMSGS')
   FETCH NEXT
      FROM tables
```

```
      INTO @tablename
END

-- Close and deallocate the cursor
CLOSE tables
DEALLOCATE tables

SELECT ObjectName, LogicalFrag
  FROM #fraglist
  WHERE LogicalFrag >= @maxfrag
    AND INDEXPROPERTY (ObjectId, IndexName, 'IndexDepth') > 0

-- Declare cursor for list of indexes to be defragged
DECLARE indexes CURSOR FOR
   SELECT ObjectName, ObjectId, IndexId, LogicalFrag
   FROM #fraglist
   WHERE LogicalFrag >= @maxfrag
      AND INDEXPROPERTY (ObjectId, IndexName, 'IndexDepth') > 0

-- Open the cursor
OPEN indexes

-- loop through the indexes
FETCH NEXT
   FROM indexes
   INTO @tablename, @objectid, @indexid, @frag

WHILE @@FETCH_STATUS = 0
BEGIN
   PRINT 'Executing DBCC INDEXDEFRAG (0, ' + RTRIM(@tablename) + ',
      ' + RTRIM(@indexid) + ') - fragmentation currently '
       + RTRIM(CONVERT(varchar(15),@frag)) + '%'
   SELECT @execstr = 'DBCC INDEXDEFRAG (0, ' + RTRIM(@objectid) + ',
       ' + RTRIM(@indexid) + ')'
   EXEC (@execstr)

   FETCH NEXT
      FROM indexes
      INTO @tablename, @objectid, @indexid, @frag
END

-- Close and deallocate the cursor
CLOSE indexes
DEALLOCATE indexes

-- Delete the temporary table

DROP TABLE #fraglist
GO
```

As well, using a filesystem level defragmentation tool on the SQL database files for WSUS can also help improve performance. I have used Piriform Defraggler (http://www.defraggler.com) with success to ensure that my WSUS database and log files are in 1 piece. (example screenshot from the "File List" tab in Defraggler 2.21.993 with a fragmented SUSDB_log.ldf file). As the WSUS database grows over time, the database files likely fragment.

☐	system_health_0_13215541901...	3	177 KB	Microsoft SQL Server Extended Event Log File	10/14/2019 12:...	D:\MSSQL11.MSSQLSERVER\MSSQL\Log\
☐	system_health_0_13215555860...	4	550 KB	Microsoft SQL Server Extended Event Log File	10/14/2019 10:...	D:\MSSQL11.MSSQLSERVER\MSSQL\Log\
☐	system_health_0_13215589897...	6	287 KB	Microsoft SQL Server Extended Event Log File	10/14/2019 10:...	D:\MSSQL11.MSSQLSERVER\MSSQL\Log\
☐	SUSDB_log.LDF	3	2,876,35...	SQL Server Database Transaction Log File	10/14/2019 10:...	D:\MSSQL11.MSSQLSERVER\Data\
☐	log_122.trc	5	1,024 KB	SQL Server Profiler - trace data file	10/14/2019 12:...	D:\MSSQL11.MSSQLSERVER\MSSQL\Log\

This log file is almost 3GB while the Database file is at almost 9GB. This WSUS server only handles ~50 PCs, but has been active for ~10 years and migrated from older Server OS versions up to Server 2016. Keep in mind that I have pruned out older updates like Windows XP, Windows 7, etc by declining those updates using SQL Scripts after changing the currently managed OSes/Applications in the WSUS Product Catalog, and then finishing up by Re-Running the WSUS Cleanup Tool.

Prologue

(revisiting this on 10/26/2019) After going through this ordeal, a few weeks ago, I returned to look at how many updates might now need cleanup in just the last few weeks (1.5 – 2 weeks) and I was surprised to see that there were 82 updates that needed cleanup… Again, based on earlier findings, these were all related to Windows Defender Security Intelligence updates.

Example: Security Intelligence Update for Windows Defender Antivirus - KB2267602 (Version 1.303.11.0) - and 81 other versions

As well, this falls in line with what the WSUS Server Cleanup Wizard showed me (albeit in different detail). I hope this helps someone other than me.