OpenText™ Application Security Database Performance and Maintenance Guide

Application Security Database Performance and Maintenance Guide
Version 26.4

PDF Generated on November 11, 2026

© Copyright 2026 Open Text

Table of Contents

Application Security Database Performance and Maintenance Guide 3
1 Change log 3
2 About this guide 3
3 Recommended database settings 4
3.1 SQL Server settings 4
3.2 Size recommendations for SQL Server databases 6
3.3 Size recommendations for MySQL Community Edition databases 6
3.4 Size recommendations for Oracle databases 7
4 Managing database performance issues 7
4.1 Disk I/O 8
4.2 Oracle databases: Automatic Workload Repository for database tuning 8
4.3 MySQL database performance tuning 8
4.4 SQL Server checks for performance tuning 9
4.5 Index fragmentation 10
4.6 OpenText Application Security Scheduler 10
4.7 Managing authentication tokens 11
4.8 Managing artifacts 11
4.8.1 Artifact questions and answers 11
4.8.2 Using scripts to delete and purge artifacts 12
4.8.3 Deleting artifacts 14
4.8.4 Purging artifacts 15
4.9 Maintenance schedule 16
4.9.1 Purge and delete script schedule 16
5 Database queries for SQL Server (on-premises) 16
5.1 Query 1: Listing SQL wait types 18
5.2 Query 2: Signal Waits (run against the main database) 20
5.3 Query 3: Information about operating system memory size and state 21
5.4 Query 4: Input/output statistics by file for the current database 21
5.5 Query 5: Volume information for all logical unit numbers with database files on the current instance 22
5.6 Query 6: Volume data for all LUNS that have database files on the current instance 23
5.7 Query 7: Drive-level latency information 23
5.8 Query 8: Index fragmentation 25
5.9 Query 9: SQL Version 25
5.10 Enable Query Store for the Application Security database 25
5.11 SQL Scripts: First Responder Kit 26

Application Security Database Performance and Maintenance Guide

- Change log
- About this guide
- Recommended database settings
- Managing database performance issues
- Database queries for SQL Server (on-premises)

1. Change log

The following table lists changes made to this document. Revisions to this document are published between software releases only if the changes made affect product functionality.

Software release / Document revision	Changes
26.4.0	Updated: • Release date and version number
26.2.0	Updated: • Release date and version number
25.4.0	Updated: • Release date and version number

2. About this guide

This guide provides guidance for users with OpenText Application Security implementations to maintain and adjust the OpenText Application Security database. This information is intended to help you understand the database solution you use with OpenText Application Security and the situations that warrant adjustments to and maintenance of your database.

OpenText strongly recommends that you work with your database administrator (DBA) to ensure that all actions are performed correctly and securely. If you do not have access to a DBA, Professional Services can help you tune and maintain your OpenText Application Security database.

Note

For information on supported databases, see the *OpenText™ Application Security Software System Requirements* document.

This document includes the following information:

- Guidance and recommendations on database hardware requirements based on the database type and metrics collected (OpenText Application Security database setting recommendations)
- Managing database performance issues provides information about:
  - Disk I/O issues related to performance (Disk I/O)
  - Indexing issues (Indexing fragmentation)
  - Scheduler options for data retention (OpenText Application Security Scheduler)
  - Managing artifacts to keep the database lean (Managing artifacts)

    - Scheduling database maintenance (Maintenance schedule)
- The appendix (Database queries: SQL Server (On Prem)) provides Microsoft® SQL Server® queries for your DBA team to execute. OpenText recommends that your DBA execute these queries.
  If you are an experienced OpenText Application Security user, and you are seeing performance issues, use these queries to collect the necessary data so that Customer Support can use it to provide feedback and recommendations.
    - Database queries: SQL Server (On Prem) provides suggestions on how to query MySQL databases.
    - Database queries: SQL Server (On Prem) provides suggestions on how to query Oracle databases.

3. Recommended database settings

The following sections provide guidance and recommendations for your OpenText Application Security database, based on database type and metrics.

- SQL Server settings
- Size recommendations for SQL Server databases
- Size recommendations for MySQL Community Edition databases
- Size recommendations for Oracle databases

3.1. SQL Server settings

The SQL Server **Cost Threshold for Parallelism** property determines the threshold at which SQL Server creates and runs parallel plans for queries. SQL Server creates and runs a parallel plan for a query only if the estimated cost of running a serial plan for the same query is higher than the value set for **Cost Threshold for Parallelism** property.



Note

The "cost" refers to the estimated cost of running the serial plan on a specific hardware configuration, and does *not* refer to a unit of time.

SQL Server Properties Security settings

OpenText recommends the following SQL Server Properties Security settings.

SQL Server Properties Security settings area	Option and recommendation
Server authentication	OpenText supports and recommends the **SQL Server and Windows Authentication mode** mixed mode for server authentication. This allows you to run OpenText Application Security in Microsoft Windows® or Linux® systems. OpenText does not recommend the **Windows Authentication mode** because this "windows only" mode limits the ability to operate in a Linux environment.
Login auditing	OpenText Application Security only uses and validates the **Failed logins only** option. However, you can choose to audit logins. OpenText does not dictate this setting. None of the options have an effect on your OpenText Application Security implementation.
Server proxy account	OpenText Application Security does not enable server proxy accounts internally. Therefore, OpenText does not provide guidance on this setting.
Options	OpenText Application Security does not select or test any of the options in this section. Therefore, OpenText does not provide guidance on these settings.

SQL Server Properties Advanced settings

OpenText recommends the following SQL Server Properties Advanced settings.

The default value for **Cost Threshold for Parallelism** is `5`, which means that the optimizer switches to a parallel plan if the cost of a query plan is more than 5. You can set this property to any value from 0 through 32767. Microsoft recommends that the property be set only by an experienced database administrator or a certified SQL Server professional.

If the **Max Degree of Parallelism** property is set to `1`, SQL Server ignores the value set for **Cost Threshold of Parallelism**. OpenText recommends that you change this value to `50`. Parallelism occurs even if you increase the **Max Degree of Parallelism** value. The aim is to minimize unwanted parallelism. SQL waits for the data to be returned from the queries that have gone parallel. In certain cases, you can choose a parallel plan, even though the query's cost plan is less than the current cost threshold for parallelism value. This can happen if the decision to use a parallel or serial plan is based on a cost estimate provided earlier in the optimization process. For information about the cost threshold for parallelism option, see https://www.brentozar.com/archive/2017/03/why-cost-threshold-for-parallelism-shouldnt-be-set-to-5.

You must verify that the **Compatibility level** setting matches the current SQL Server engine version. Typically, when users upgrade to a newer SQL Server version, they back up and restore the OpenText Application Security database to a new SQL Server that is running the latest supported SQL Server version. When you restore the OpenText Application Security database, the compatibility level is set to the SQL Server version on which the OpenText Application Security database was previously based.

The following images show an example of the properties for a OpenText Application Security database that was restored to SQL Server 2019.


SQL Server database compatibility level set to SQL Server 2008

You must change the **Compatibility level** setting to reflect the current SQL Server engine version.


SQL Server database compatibility level set to SQL Server 2019

3.2. Size recommendations for SQL Server databases

These recommendations are based on a limited data set and must not be construed as definitive. Their usability should increase as more data becomes available. The following table lists the size recommendations for OpenText Application Security instances that use the SQL Server database.

If the database size reaches 5 TB, OpenText recommends that you deploy multiple OpenText Application Security servers.

	Small	Medium	Large	Extra large
Data size (TB)	1	1–3	3–5	5–10
Scans per week	up to 5 K	5–15 K	15–25 K	25+ K
Number of users	up to 1 K	1–5 K	5–10 K	10+ K
Number of application versions	up to 5 K	5–20 K	20–50 K	50+ K
RAM (GB)	64	128	256	512
Processors	8-core	16-core	32-core	64-core
IOPS (minimum)	32,000	32,000+	64,000	64,000+
Cost Threshold for Parallelism[1]	50	50	50	50

[1] The value must be a minimum of 50 and can be increased based on evaluated performance needs.

3.3. Size recommendations for MySQL Community Edition databases

Important

OpenText does not recommend using MySQL Community Edition databases for large enterprise implementations. If the database size reaches 500 GB, OpenText recommends that you use the MySQL Enterprise Edition database.

The following table lists the size recommendations for OpenText Application Security instances that use MySQL Community Edition databases.

	Small	Medium	Large	Extra large
Data size (TB)	1	1–3	3–5	5–10
Scans per week	up to 5 K	-	-	-
Number of users	up to 1 K	-	-	-
Number of application versions	up to 5 K	-	-	-
RAM (GB)	64	-	-	-
Processors	8-core	-	-	-
IOPS (minimum)	32,000	-	-	-
Cost Threshold for Parallelism	50	-	-	-

3.4. Size recommendations for Oracle databases

The following table lists the size recommendations for OpenText Application Security instances that use the Oracle® Database.

If the database size reaches 5 TB, OpenText recommends that you deploy multiple OpenText Application Security servers.

	Small	Medium	Large	Extra large
Data size (TB)	1	1–3	3–5	5–10
Scans per week	up to 5 K	5–15 K	15–25 K	25,000+
Number of users	up to 1 K	1–5 K	5–10 K	10,000+
Number of application versions	up to 5 K	5–20 K	20–50 K	50,000+
RAM (GB)	64	128	256	512
Processors	8-core	16-core	32-core	64-core
IOPS (minimum)	32,000	32,000+	64,000	64,000+

4. Managing database performance issues

The following sections provide guidance to address performance issues related to disk I/O, indexing, scheduler options for data retention, managing artifacts, and scheduling database maintenance.

- Disk I/O
- Oracle databases: Automatic Workload Repository for database tuning
- MySQL database performance tuning
- SQL Server checks for performance tuning
- Index fragmentation
- OpenText Application Security Scheduler
- Managing authentication tokens
- Managing artifacts

- Maintenance schedule

4.1. Disk I/O

Disk I/O encompasses the input/output operations on a physical disk. In reading data from a file on a disk, the processor must wait for the file to be read (the same applies to writing data to a file). OpenText Application Security is a high I/O-intensive application, which affects performance.

Performance begins to degrade after the OpenText Application Security database reaches a certain size. Monitoring overall read/write latency is especially important as data volume approaches 1 TB. At that point, the database deployment requires database adjustments.

OpenText Application Security application workloads tend to grow rapidly in total data, size of the active data set, and the compute needed to satisfy growing transaction requirements.



Note

Table cleanups that use purge or delete of artifacts or application versions might not result in actual reduction of database storage allocation until the database administrator re-optimizes the database. OpenText recommends regular monitoring and optimization of the OpenText Application Security databases.

Amazon RDS provides three storage types, which are described on the Amazon Relational Database Service documentation website.

- General Purpose SSD – General Purpose SSD volumes offer cost-effective storage that is ideal for a broad range of workloads running on medium-sized database instances. General Purpose storage is best suited for development and testing environments. For more information about General Purpose SSD storage, including the storage size ranges, see **General Purpose SSD storage** on the Amazon Relational Database Service Documentation website (https://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/CHAP_Storage.html).
- Provisioned IOPS SSD – Provisioned IOPS storage is designed to meet the needs of I/O-intensive workloads, particularly database workloads, that require low I/O latency and consistent I/O throughput. Provisioned IOPS storage is best suited for production environments. For more information about Provisioned IOPS storage, including the storage size ranges, see **Provisioned IOPS SSD storage** on the Amazon Relational Database Service Documentation website (https://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/CHAP_Storage.html).
- Magnetic – Amazon RDS supports magnetic storage for backward compatibility. Amazon recommends that you use General Purpose SSD or Provisioned IOPS SSD for any new storage needs. The maximum amount of storage allowed for database instances on magnetic storage is less than that of the other storage types. For more information, see **Magnetic storage** on the Amazon Relational Database Service Documentation website (https://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/CHAP_Storage.html).

4.2. Oracle databases: Automatic Workload Repository for database tuning

The Automatic Workload Repository (AWR) report includes data on database activity between two points in time. The reports contain a large amount of database performance data. You can use this information to compare statistics captured during a period of poor performance to a baseline, and then diagnose performance issues.

OpenText recommends that, before you contact Customer Support for help with a OpenText Application Security performance issue, you first generate an AWR report and have your Oracle DBA review any recommendations it provides. For detailed information about AWR, see the Oracle Database Documentation.

4.3. MySQL database performance tuning

The MySQL Percona Toolkit offers a toolkit that includes the pt-diskstats tool, which you can use to monitor disk I/O for MySQL databases. The pt-diskstats tool is similar to iostat but is more interactive and detailed. For more information on the pt-diskstats tool, see Percona Toolkit - pt-diskstats.

You can use MySQL Enterprise Monitor to help analyze and tune your database performance.

MySQL InnoDB buffer pool is an in-memory cache between InnoDB pages on disk and MySQL query runs. When MySQL reads a page, it first checks if the page exists in the InnoDB buffer pool. MySQL skips the disk I/O operation and reads the

pages from the buffer pool. If the pages do not exist in the buffer pool, MySQL retrieves the pages from the disk, adds the pages to the buffer pool, and then continues to run the query.

The buffer pool enhances the performance of MySQL query runs. Otherwise, a query overload uses more disk I/O operations.

> **Note**
>
> OpenText recommends that you set the MySQL InnoDB buffer pool to 75% of the total memory.

4.4. SQL Server checks for performance tuning

Customer support provides several SQL queries that you can use to gather data directly from SQL Server using dynamic management views. Dynamic management views and functions return server state information that you can use to monitor the health of a server instance, diagnose problems, and tune performance. To run these queries, you must have SQL Server sysadmin rights. OpenText recommends that you follow SQL Server best practices.

SQL Server Wait Statistics

One of the most under-used performance troubleshooting methodologies in the SQL Server world is *Wait Statistics*, or simply *wait stats*. The wait stats performance object contains performance counters that report information about broad categorizations of waits. You can identify the performance issue by analyzing the wait stats, and then use the results to see where to take the necessary actions to resolve the issue.

SQL Server "knows" where performance issues exist. One such issue is CPU Pressure. Signal waits above 10-15% often indicate a CPU pressure issue.

> **Note**
>
> For detailed information about Wait Statistics, see SQL Server, Wait Statistics object" in Microsoft Learn.

The following results, returned for a query for read/write latency (see Query 7: Drive-level latency information) shows results for the E:\ drive, which is where the OpenText Application Security database is located.


Results of SQL Server query 7 for read and write latency

The following table is a good reference against which to compare your read/write latency values.

Latency value / Value range (ms)	Interpretation
> 1	Excellent
> 5	Very good
5–10	Good
10–20	Poor
20–100	Bad
100–500	Very bad
> 500	Ridiculously bad

Checks for a SQL Server database

If you are using a SQL Server database as the OpenText Application Security database, perform the following checks:

- Enable the Auto Update Stats Asynchronously (`AUTO_UPDATE_STATISTICS_ASYNC`) option for the database. For instructions, see the Microsoft SQL Server documentation website.

- Make sure that your SQL Server database schema collation is case-sensitive. The default installation of SQL Server is case-insensitive.

  Important

  Before you run the OpenText-provided SQL scripts, verify that there are no open connections to the database.

- Ensure that snapshot isolation is enabled (`ALLOW_SNAPSHOT_ISOLATION` and `READ_COMMITTED_SNAPSHOT` are set to ON) on the database schema used for the installation.
- During SQL script executions, check the client tool to make sure that its ANSI null default option is set to ON. To do this, you can either use a `SET` command (set `ANSI_NULL_DFLT_ON` to ON) or the Query Editor.

4.5. Index fragmentation

Index fragmentation is a common source of database performance degradation. Fragmentation occurs when there is a lot of empty space on a data page (internal fragmentation) or when the logical order of pages in the index does not match the physical order of pages in the data file (external fragmentation).

The following table lists tools you can use to address fragmentation in each supported database.

Database	Tool
SQL Server	Microsoft has its own maintenance solution, which is AdaptiveIndexDefrag. AdaptiveIndexDefrag performs an intelligent defrag on one or more indexes, for one or more databases. For details on what AdaptiveIndexDefrag does and how to use it, see Adaptive Index Defrag.
	You can use the SQL Server Index and Statistics Maintenance Solution. For details about this solution and how to use it, see SQL Server Index and Statistics Maintenance.
Oracle	Oracle provides a script to check for index fragmentation in its databases. For information, see Check for Index Fragmentation. Be aware, however, that running this script takes locks out of the indexes.
MySQL	For information on how to check for index fragmentation in MySQL databases, see the topic Check for Fragmentation in MySQL and fix it.

4.6. OpenText Application Security Scheduler

If you allow artifacts and application versions to grow over time without maintenance, you might begin to see performance issues and extended upgrade times between releases. Users typically notice the degraded performance when their OpenText Application Security database size reaches 1 TB. This can easily happen if you have many years of unpurged data.

You can configure a data retention policy that enables you to define the time period for which artifacts are retained. You can configure advanced options to define the time period to retain the artifacts and the number of artifacts to retain per application version. After the defined retention time period is reached, the artifacts are eligible for purging. You can schedule the cleanup service to purge artifacts. For more information, see the *OpenText™ Applicaton Security User Guide*.

If your OpenText Application Security version is earlier than version 24.2.0, Customer Support provides a set of PowerShell scripts that you can use to purge or delete artifacts. These scripts enable you to download the artifacts before you purge or

delete them. The goal is to keep the database as trim and performant as possible.

Along with the data retention policy, OpenText Application Security provides the following three data retention settings (under **Administration > Configuration > Scheduler**):

- Events maintenance
  Use the **Events maintenance** > **Days to preserve** setting to specify the number of days after which OpenText Application Security removes past events. The default is zero (0), which results in no event removal.
  Consider configuring this setting to 35 days. Anything longer might result in the addition of millions of rows to the `dbo.eventlogentry` table, which stores all events that you see in the OpenText Application Security user interface.
  To back up all existing events, you can export the event logs in OpenText Application Security (**Administration > Metrics & Tracking > Event Logs**). After you do, you can safely truncate the `dbo.eventlogentry` table. OpenText recommends that you truncate the table on a regular basis.
- Reports maintenance
  Use the **Reports maintenance > Days to preserve** setting to automatically delete reports that are no longer required after the specified number of days.

  > **Note**
  > If you are using OpenText Application Security 22.1.x or earlier versions, Customer Support can provide a PowerShell script that you can run to create a list of generated reports, which you can then use to delete reports no longer required.

- Data exports maintenance
  Use the **Data exports maintenance > Days to preserve** setting to regularly remove data users have exported. By default, this is configured to 2 days. OpenText recommends that you leave the default setting of `2`.

4.7. Managing authentication tokens

There is no data retention policy for expired OpenText Application Security authentication tokens. OpenText recommends that administrators periodically review the list of authentication tokens, and delete the expired tokens. For information about how to delete authentication tokens, see the *OpenText™ Applicaton Security User Guide*.

4.8. Managing artifacts

An artifact in OpenText Application Security is a container for various types of content. The most important artifacts are Fortify Project Results (FPR) files. Typically, an FPR contains one scan produced by a specific analyzer such as OpenText™ Fortify Static Application Security Testing (OpenText Fortify SAST) or OpenText™ Fortify Dynamic Application Security Testing (OpenText Fortify DAST), but can include multiple scans produced by different analyzers. When an FPR file is uploaded to OpenText Application Security, the corresponding artifact gets a system date stamp as the artifact upload date. A scan inside of an artifact has a scan date apart from the artifact upload date.

For simplicity, assume that:

- Each of several artifacts uploaded to OpenText Application Security contains just one FPR file.
- The artifacts were uploaded in the same order that the scans were performed. So, for example, the FPR from the oldest scan is uploaded first.

4.8.1. Artifact questions and answers

Q: On a new OpenText Application Security version, I can add multiple OpenText Fortify SAST artifacts, and then delete any of them. How does the state handle this?

A: When you delete any of the artifacts, OpenText Application Security recalculates the new state based on the remaining artifacts. When the last artifact associated with an application version is deleted, the application version acquires “no state” (it becomes a completely empty application version).

Q: If I have five artifacts and I purge numbers 6 through 10, why is it possible to delete numbers 1 through 4, but not 5? That makes no sense to me.

A: Issues are stored in two places, one for scan issues storage and one for issue storage. *Scan issue storage* holds issues associated with each uploaded scan. This storage contains only issues present in a specific scan. *Issue storage* contains the current issues state for each application version.

> *The following example illustrates the difference:*
>
> *Scan 1 contains issue A and Scan 2 contains issues A and B.*
>
> *In this example, Scan issue storage contains three issues: one for Scan 1 (issue A) and two for Scan 2 (issues A and B). Issue storage contains two issues: A (the status of which is UPDATED) and B (the status of which is NEW).*
>
> *This separation is needed to simplify and speed up the querying of current issues state. OpenText Application Security does not need to analyze all the scan issues to calculate issue status at the moment of query. It just selects precalculated current issues from issue storage.*
>
> *The separation would be unnecessary if OpenText Application Security used the latest scan result as the issue state. But, in addition to tracking when the issue was updated (found by the current and previous scans), OpenText Application Security also tracks removed and reintroduced issues. Because of this, having scan issues for previous scans is important for current scan processing and final issue state calculation.*
>
> *Issue storage state must be recalculated every time a new scan is uploaded. Scan issue storage is the primary source of information used for issue state calculation.*
>
> *A purge operation removes data from scan issue storage to decrease the amount of disc space the database uses. Note that purging does not delete data from current issues storage and does not affect the current application version issues state. So, if you purged* `v3.fpr` *, you would see the following:*
>
> 
> *Artifact history page that shows v3.fpr, v2.fpr, and v1.fpr as purged*
>
> *OpenText Application Security now contains full scan data related to scans 6, 5, and 4 (scan and scan issues). Scan results associated with scans 3, 2 and 1 are removed. After that happens, OpenText Application Security cannot allow the deletion of any purged scans because scan results associated with these scans are gone and OpenText Application Security cannot reliably recalculate the state of all issues whose status is not NEW, but REMOVED, REINTRODUCED and UPDATED.*
>
> ***Notes on artifact/scan ordering***
>
> *Even if you can upload artifacts in an order that is different from the order of scan dates, OpenText strongly recommends against it, especially if you want to use purging. The visual representation that would result in the* ***ARTIFACT HISTORY*** *table would be confusing. There is no scan date column in the table because of the 1:N relation between FPRs and scans.*

Q: Why is the analysis date used to determine the issues shown instead of the upload date?

A: This is the only reliable way to track issue status. If the upload date was used to calculate the current issues state, the results would be inconsistent.

> *For example:*
>
> *Scan 1 uncovered one issue in the application source code. This issue was fixed, and scan 2 uncovered no issues. The order in which the results of these two scans are uploaded does not matter. The issue status is always removed after both scans are uploaded. If OpenText Application Security used the upload date for scan calculation, it would not be true. If scan 1 was uploaded before scan 2, the issue status would be REMOVED. But if scan 2 is uploaded first, followed by scan 1, the issue status would be incorrectly set to NEW, which is wrong. The issue that no longer exists in code would be marked as an active, new issue.*

4.8.2. Using scripts to delete and purge artifacts

Customer Support provides the following set of PowerShell scripts that you can use to either purge or delete artifacts:

- `configFile.ps1`
- `GenerateListofArtifacts.ps1`
- `DeleteArtifacts.ps1`
- `PurgeArtifacts.ps1`
- `ConnectivityTestToSSC.ps1`

These scripts enable you to download artifacts and then perform the purge or delete so that the database is kept as trim as possible. If you use the scripts to download the artifacts, they create a parent directory based on the application name, and a sub-directory based on the application version. The downloaded artifacts are placed in the output directory that you specify in the `configFile.ps1` script.

The `ConnectivityTestToSSC.ps1`, which you run first, is designed to test connectivity between the machine from which you run the PowerShell scripts and OpenText Application Security. The script attempts to connect to OpenText Application Security, then requests a UnifiedLoginToken.

Before you run any of the PowerShell scripts, you must first verify the PowerShell Execution Policy on Windows and connect to OpenText Application Security by doing the following:

1. Run the following command using Windows PowerShell ISE as an administrator:

   ```
   get-executionpolicy
   ```

   To set the policy to unrestricted, run:

   ```
   set-executionpolicy unrestricted
   ```

   <img data-zoom="true"
   src="file/images/general/fsscdb/SSC_DB_Perf_Proj/ssc_db_perf_proj_imagesGet_Exec_Policy.png" style="border: none !important;"/>
2. Use PowerShell ISE run as an administrator) to run the `ConnectivityTestToSSC.ps1` script, as shown here:
   <img data-zoom="true"
   src="file/images/general/fsscdb/SSC_DB_Perf_Proj/ssc_db_perf_proj_imagesConnectivity.png" style="border: none !important;"/>
3. Provide credentials when prompted.
   The information you enter is not saved.
   <img data-zoom="true"
   src="file/images/general/fsscdb/SSC_DB_Perf_Proj/ssc_db_perf_proj_imagesPleaseRead.png" style="border: none !important;"/>
   After the script successfully connects to OpenText Application Security, it directs OpenText Application Security to create a UnifiedLoginToken authentication token, and then revokes that token.
   <img data-zoom="true"
   src="file/images/general/fsscdb/SSC_DB_Perf_Proj/ssc_db_perf_proj_imagesRevokedToken.png" style="border: none !important;"/>

Next, run the `configFile.ps1` and `GenerateListofArtifacts.ps1` scripts, as follows:

1. Using Windows PowerShell ISE, run `configFile.ps1`.
   ssc_db_perf_proj_imagesConfigFile_ps1
2. Provide values for the `$sscBaseURL` and `$outputDirectory` variables.
3. Run `GenerateListofArtifacts.ps1`.
4. Review the information displayed.
   The `GenerateListofArtifacts.ps1` script generates the following text files and places them in the directory you specified for the `$outputDirectory` variable:
   - A list of artifacts that can be purged.
   - A list of artifacts that can be deleted.
   - A list of application versions with no associated artifacts
   - A list of artifacts with the "Purging" status.

5. Respond to the following prompts:
    - Did you READ the above statements (Yes):
    - Enter a valid date to find artifacts older than the date provided. If you wish to do a full audit, enter tomorrow's date. (MM/DD/YYYY):
    - Please enter SSC Account Name (assigned Admin Role):
    - Please enter Password for SSC Account admin:

The `GenerateListofArtifacts.ps1` script generates the following text files and places them in the directory you specified for the `$outputDirectory` variable:

- `ArtifactsPurgeList.txt` lists the artifacts that can be purged.
  The `PurgeArtifacts.ps1` script uses this file content as its input. For example:
  example contents of the ArtifactsPurgeList.txt file
- `ArtifactsDeleteList.txt` lists the artifacts that can be deleted.
  The first column displays the total number of artifacts associated with the application version. The `DeleteArtifacts.ps1` script uses this file content as its input. For example:
  example contents of the ArtifactsDeleteList.txt file
- `SSCProjectsWithNoArtifacts.txt` lists application versions that have no associated artifacts.
  The `href` can be used to submit a `POSTDelete` to delete each application version with no associated artifacts. For example:
  example contents of the SSCProjectsWithNoArtifacts.txt file
- `ArtifactsWithPurging_Status.txt` lists artifacts with the "Purging" status. For example:
  example contents of the ArtifactsWithPurging_Status.txt file
  If you have artifacts that are stuck in 'Purging' status, contact Customer Support for assistance.
- `SummaryReport_Deletes.txt` file displays the number of artifacts that can be deleted, and the number that would remain after deletion is completed. For example:
  example contents of the SummaryReport_Deletes.txt file
  If you decide to delete artifacts, this file gives you an idea of how many artifacts could be deleted and how many would be left. If, after deletion, an application version would have zero artifacts, you can just delete it from the `ArtifactsDeleteList.txt` file.

4.8.3. Deleting artifacts

To delete artifacts:

1. Run the `DeleteArtifacts.ps1` script.
   The following information is displayed:

```
************************ IMPORTANT ************************
Based on the date provided (when you involved the GenerateListofArtifacts.ps1), artifacts have
been found that could be DELETED.
Please review the file 'D:/Powershell script/Purge_Delete Scripts/output/ArtifactsDeleteList.txt'.
The generated file will be used by the DeleteArtifacts.ps1 script.
        ****Delete operation will revert all traces of an artifact!****
           ****Application history will be impacted!****
RECOMMENDATION: If you are planning to DELETE the artifacts no need to download.
The Delete script will prompt you if you wish to download the artifacts.
RECOMMENDATION: DELETE in batches.
***********************************************************
```

   The delete operation removes all traces of an artifact. Application history is affected, but you free up more storage in the OpenText Application Security database.

Important

For every purge request, a corresponding purge job is created in OpenText Application Security. OpenText strongly recommends that you purge artifacts in batches of no more than 100 at a time. You can monitor the corresponding Delete jobs to track performance and potentially increase the number of artifacts per batch.

2. Respond to the following prompts.
   You can tell the script to download the artifacts before their deletion. The credentials you provide to access OpenText Application Security are not saved.

```
Did you READ the above Statements (Yes)?:
The user has acknowledged that the statements have been reviewed.
Confirming that you wish to DELETE the artifacts (Yes)?:
The user has acknowledged that they wish to DELETE the artifacts (listed in 'D:/Powershell
script/Purge_Delete Scripts/output/ArtifactsDeleteList.txt'). An SSC Artifact 'Delete job' will be
created for each artifact. You can exit the script now to change your option.
Artifacts with status of 'ERROR PROCESSING' will be ignored for downloading. Download Artifacts
before the 'DELETE' operation occurs (Yes/No)?:
The user has acknowledged that the artifact(s) will be downloaded prior to the 'DELETE' operation.
Please Enter SSC Account Name (Assigned Admin Role):
Please Enter Password for SSC Account admin:
```

4.8.4. Purging artifacts

The purge operation removes artifacts from the system and recovers space in the database without affecting issue metrics. The database space reclaimed is not as extensive as that reclaimed by the delete artifact operation.

To purge artifacts:

1. Run the `PurgeArtifacts.ps1` script.
   The following Information is displayed:

```
************************ IMPORTANT ************************
Based on the date provided (when you invoked the GenerateListofArtifacts.ps1), artifacts have been
found that could be PURGED.
Please review the file 'D:/Powershell script/Purge_Delete Scripts/output/ArtifactsPurgeList.txt'.
The generated file will be used by the PurgeArtifacts.ps1 script.
        ****Purging artifacts from the system recovers space without affecting issue metrics****
           ****Application history will be impacted!****
RECOMMENDATION: If you are planning to PURGE the artifacts we recommend that you download them.
The Purge script will prompt you if you wish to download the artifacts.
RECOMMENDATION: PURGE in batches.
************************************************************
```

Important

For every purge request there is a corresponding purge job created in OpenText Application Security. OpenText strongly recommends that you purge artifacts in batches of no more than 100 at a time. You can monitor the corresponding Delete jobs to track performance and potentially increase the number of artifacts per batch.

2. Respond to the prompts as the following content is displayed.
   You can direct the script to download the artifacts before deleting them. The account and password used to access OpenText Application Security are not saved.

```
Did you READ the above Statements (Yes)?:
The user has acknowledged that the statements have been reviewed.
Confirming that you wish to PURGE the artifacts (Yes)?:
The user has acknowledged that they wish to purge the artifacts (listed in 'D:/Powershell script/
Purge_Delete Scripts/output/ArtifactsPurgeList.txt'). An SSC Artifact 'Purge job' will be created
for each artifact. You can exit the script now to change your option.
Artifacts with the Status of 'ERROR PROCESSING' or 'REQUIRE AUTH' will be ignored for downloading.
Download Artifacts before the 'PURGE' operation occurs (Yes/No)?:
The user has acknowledged that the artifact(s) will be downloaded prior to the 'PURGE' operation.
Please Enter SSC Account Name (Assigned Admin Role):
Please Enter Password for SSC Account admin:
```

4.9. Maintenance schedule

A *database* is designed to make transactional systems run efficiently. Typically, it is an online transaction processing (OLTP) database, which is usually constrained to a single application.

A *data warehouse* is a different kind of database. A data warehouse exists as a layer on top of another database or databases (usually OLTP databases). The data warehouse takes the data from all these databases and creates a layer optimized for and dedicated to analytics. OpenText Application Security is not designed as a data warehouse.

Keeping 10 years' worth of data in OpenText Application Security is not practical unless you deploy more than one instance. If deploying multiple OpenText Application Security instances is not an option, OpenText recommends that you keep no more than two years of data for optimal performance.

For an application version that is no longer active or needed, you can download the latest merged scan results. For instructions, see the *OpenText™ Applicaton Security User Guide*. After you download the data, you can delete the application version. If you need to access the information in the FPR, upload it to a non-production OpenText Application Security server.

4.9.1. Purge and delete script schedule

OpenText recommends that you run the analysis, purge, and delete queries every six months, or sooner if you observe performance issues. If your database is over 1 TB, OpenText recommends that you analyze and clean your database quarterly.

If you have several years of data, and you want to maintain only two years of data, you can use the purge or delete scripts to purge or delete older artifacts.

The delete operation removes all traces of an artifact and the application version history is affected, and you reclaim storage space in the database. The purge operation also enables you to reclaim space in the database, but less than the delete operation, because issue history is maintained.

5. Database queries for SQL Server (on-premises)

This section provides lists of queries for SQL Server (on-premises) databases. These queries use the dynamic management views shipped with SQL Server. OpenText recommends that your database administrator execute these queries.

If you are a new OpenText Application Security user, you must run these queries to establish a baseline. As your database grows, and approaches 1 TB in size, consider re-running the queries and comparing the data to the baseline data.

If you are an experienced OpenText Application Security user, and you are seeing performance issues, use these queries to collect the necessary data so that Customer Support can use it to provide feedback and recommendations.

An output example and a description of what to look for are provided for each query.

Note

Queries for MySQL and Oracle database types, if applicable, will be added in a future release.

- Query 1: Listing SQL wait types
- Query 2: Signal Waits (run against the main database)
- Query 3: Information about operating system memory size and state
- Query 4: Input/output statistics by file for the current database
- Query 5: Volume information for all logical unit numbers with database files on the current instance
- Query 6: Volume data for all LUNS that have database files on the current instance
- Query 7: Drive-level latency information
- Query 8: Index fragmentation
- Query 9: SQL Version
- Enable Query Store for the Application Security database
- SQL Scripts: First Responder Kit

5.1. Query 1: Listing SQL wait types

The following query, run against the main database, generates a list of SQL wait types:

```
IF OBJECT_ID('tempdb..#ignorable_waits') IS NOT NULL

DROP TABLE #ignorable_waits;

GO

create table #ignorable_waits (wait_type nvarchar(256) PRIMARY KEY);

GO

/* We aren't using row constructors to be SQL 2005 compatible */

set nocount on;

insert #ignorable_waits (wait_type) VALUES ('REQUEST_FOR_DEADLOCK_SEARCH');

insert #ignorable_waits (wait_type) VALUES ('SQLTRACE_INCREMENTAL_FLUSH_SLEEP');

insert #ignorable_waits (wait_type) VALUES ('SQLTRACE_BUFFER_FLUSH');

insert #ignorable_waits (wait_type) VALUES ('LAZYWRITER_SLEEP');

insert #ignorable_waits (wait_type) VALUES ('XE_TIMER_EVENT');

insert #ignorable_waits (wait_type) VALUES ('XE_DISPATCHER_WAIT');

insert #ignorable_waits (wait_type) VALUES ('FT_IFTS_SCHEDULER_IDLE_WAIT');

insert #ignorable_waits (wait_type) VALUES ('LOGMGR_QUEUE');

insert #ignorable_waits (wait_type) VALUES ('CHECKPOINT_QUEUE');

insert #ignorable_waits (wait_type) VALUES ('BROKER_TO_FLUSH');

insert #ignorable_waits (wait_type) VALUES ('BROKER_TASK_STOP');

insert #ignorable_waits (wait_type) VALUES ('BROKER_EVENTHANDLER');

insert #ignorable_waits (wait_type) VALUES ('SLEEP_TASK');

insert #ignorable_waits (wait_type) VALUES ('WAITFOR');

insert #ignorable_waits (wait_type) VALUES ('DBMIRROR_DBM_MUTEX')

insert #ignorable_waits (wait_type) VALUES ('DBMIRROR_EVENTS_QUEUE')

insert #ignorable_waits (wait_type) VALUES ('DBMIRRORING_CMD');

insert #ignorable_waits (wait_type) VALUES ('DISPATCHER_QUEUE_SEMAPHORE');

insert #ignorable_waits (wait_type) VALUES ('BROKER_RECEIVE_WAITFOR');

insert #ignorable_waits (wait_type) VALUES ('CLR_AUTO_EVENT');

insert #ignorable_waits (wait_type) VALUES ('DIRTY_PAGE_POLL');
```

```
insert #ignorable_waits (wait_type) VALUES ('HADR_FILESTREAM_IOMGR_IOCOMPLETION');

insert #ignorable_waits (wait_type) VALUES ('ONDEMAND_TASK_QUEUE');

insert #ignorable_waits (wait_type) VALUES ('FT_IFTSHC_MUTEX');

insert #ignorable_waits (wait_type) VALUES ('CLR_MANUAL_EVENT');

insert #ignorable_waits (wait_type) VALUES ('SP_SERVER_DIAGNOSTICS_SLEEP');

insert #ignorable_waits (wait_type) VALUES ('QDS_CLEANUP_STALE_QUERIES_TASK_MAIN_LOOP_SLEEP');

insert #ignorable_waits (wait_type) VALUES ('QDS_PERSIST_TASK_MAIN_LOOP_SLEEP');

GO

/* Want to manually exclude an event and recalculate?*/

/* insert #ignorable_waits (wait_type) VALUES (''); */

/*********************************

What are the highest overall waits since startup?

*********************************/

SELECT TOP 25

os.wait_type,

SUM(os.wait_time_ms) OVER (PARTITION BY os.wait_type) as sum_wait_time_ms,

CAST(

100.* SUM(os.wait_time_ms) OVER (PARTITION BY os.wait_type)

/ (1. * SUM(os.wait_time_ms) OVER () )

AS NUMERIC(12,1)) as pct_wait_time,

SUM(os.waiting_tasks_count) OVER (PARTITION BY os.wait_type) AS sum_waiting_tasks,

CASE WHEN SUM(os.waiting_tasks_count) OVER (PARTITION BY os.wait_type) > 0

THEN

CAST(

SUM(os.wait_time_ms) OVER (PARTITION BY os.wait_type)

/ (1. * SUM(os.waiting_tasks_count) OVER (PARTITION BY os.wait_type))

AS NUMERIC(12,1))

ELSE 0 END AS avg_wait_time_ms,

CURRENT_TIMESTAMP as sample_time
```

```
FROM sys.dm_os_wait_stats os

LEFT JOIN #ignorable_waits iw on

os.wait_type=iw.wait_type

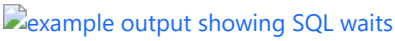
WHERE

iw.wait_type is null

ORDER BY sum_wait_time_ms DESC;

GO
```

Example output (SQL waits will vary)


example output showing SQL waits

The following three SQL waits are useful for finding disk I/O bottlenecks and for making sure that the `READ_COMMITTED_SNAPSHOT` database option is enabled on the OpenText Application Security database:

- The *LCK_M_S wait* occurs if a request is waiting to acquire a shared lock. This typically happens when read requests are blocked by write transactions (implicit or explicit) that have been kept open for extended periods of time.
- The *LCK_M_X wait* occurs if a transaction is waiting to acquire an exclusive lock in order to modify data. This lock prevents other transactions from accessing the objects, so no other processes can read or modify data.
  Lock waits commonly occur on busy servers where concurrent transactions demand the same resource, resulting in poor performance. A high number of locking waits may indicate blocking problems and should be investigated.

  Note
  If you see `LCK_M_X` , `LCK_M_IX` in the list, make sure that the `READ_COMMITTED_SNAPSHOT` option is enabled.

- The *LCK_M_U wait* occurs while a request is waiting to acquire an update lock. An update lock is not just for UPDATE operations. It is used when SQL Server needs to read, and then modify, a row, page, or table. Before SQL Server makes any changes, it places an update lock on the data. Once the system is ready, these locks are upgraded to exclusive locks. This wait typically occurs while modify requests are blocked by other write transactions (implicit or explicit).

Note
If you see `CXPACKET` in the list of SQL wait types, see the article Why Cost Threshold For Parallelism Shouldn't Be Set To 5.

5.2. Query 2: Signal Waits (run against the main database)

Signal waits indicate possible internal CPU pressure. The CPU signal waits percent is a ratiometric that compares signal waits to total waits, as a percent. That means you can see a spike in signal waits from one minute to the next without the server itself showing high CPU usage.

```
SELECT CAST(100.0 * SUM(signal_wait_time_ms) / SUM (wait_time_ms) AS NUMERIC(20,2)) AS [% Signal (CPU)
Waits],
CAST(100.0 * SUM(wait_time_ms - signal_wait_time_ms) / SUM (wait_time_ms) AS NUMERIC(20,2)) AS [%
Resource Waits]
FROM sys.dm_os_wait_stats WITH (NOLOCK)
WHERE wait_type NOT IN (
        N'BROKER_EVENTHANDLER', N'BROKER_RECEIVE_WAITFOR', N'BROKER_TASK_STOP',
              N'BROKER_TO_FLUSH', N'BROKER_TRANSMITTER', N'CHECKPOINT_QUEUE',
        N'CHKPT', N'CLR_AUTO_EVENT', N'CLR_MANUAL_EVENT', N'CLR_SEMAPHORE',
        N'DBMIRROR_DBM_EVENT', N'DBMIRROR_EVENTS_QUEUE', N'DBMIRROR_WORKER_QUEUE',
              N'DBMIRRORING_CMD', N'DIRTY_PAGE_POLL', N'DISPATCHER_QUEUE_SEMAPHORE',
        N'EXECSYNC', N'FSAGENT', N'FT_IFTS_SCHEDULER_IDLE_WAIT', N'FT_IFTSHC_MUTEX',
        N'HADR_CLUSAPI_CALL', N'HADR_FILESTREAM_IOMGR_IOCOMPLETION', N'HADR_LOGCAPTURE_WAIT',
              N'HADR_NOTIFICATION_DEQUEUE', N'HADR_TIMER_TASK', N'HADR_WORK_QUEUE',
        N'KSOURCE_WAKEUP', N'LAZYWRITER_SLEEP', N'LOGMGR_QUEUE', N'ONDEMAND_TASK_QUEUE',
        N'PWAIT_ALL_COMPONENTS_INITIALIZED', N'QDS_PERSIST_TASK_MAIN_LOOP_SLEEP',
        N'QDS_CLEANUP_STALE_QUERIES_TASK_MAIN_LOOP_SLEEP', N'REQUEST_FOR_DEADLOCK_SEARCH',
              N'RESOURCE_QUEUE', N'SERVER_IDLE_CHECK', N'SLEEP_BPOOL_FLUSH', N'SLEEP_DBSTARTUP',
              N'SLEEP_DCOMSTARTUP', N'SLEEP_MASTERDBREADY', N'SLEEP_MASTERMDREADY',
        N'SLEEP_MASTERUPGRADED', N'SLEEP_MSDBSTARTUP', N'SLEEP_SYSTEMTASK', N'SLEEP_TASK',
        N'SLEEP_TEMPDBSTARTUP', N'SNI_HTTP_ACCEPT', N'SP_SERVER_DIAGNOSTICS_SLEEP',
              N'SQLTRACE_BUFFER_FLUSH', N'SQLTRACE_INCREMENTAL_FLUSH_SLEEP', N'SQLTRACE_WAIT_ENTRIES',
              N'WAIT_FOR_RESULTS', N'WAITFOR', N'WAITFOR_TASKSHUTDOWN', N'WAIT_XTP_HOST_WAIT',
              N'WAIT_XTP_OFFLINE_CKPT_NEW_LOG', N'WAIT_XTP_CKPT_CLOSE', N'XE_DISPATCHER_JOIN',
        N'XE_DISPATCHER_WAIT', N'XE_TIMER_EVENT') OPTION (RECOMPILE);
```

Example output


Example output of signal waits

Signal waits that exceed 10-15% are typically a sign of CPU pressure.

- Cumulative wait stats are not as useful on an idle instance that is not under load or performance pressure
- Resource waits are non-CPU-related waits

For information about how to troubleshoot high CPU usage issues in SQL Server, see Troubleshoot high-CPU-usage issues in SQL Server.

5.3. Query 3: Information about operating system memory size and state

Run the following query against the main database to generate basic information about your operating system memory size and state:

```
SELECT total_physical_memory_kb/1024 AS [Physical Memory (MB)],
       available_physical_memory_kb/1024 AS [Available Memory (MB)],
       total_page_file_kb/1024 AS [Total Page File (MB)],
          available_page_file_kb/1024 AS [Available Page File (MB)],
          system_cache_kb/1024 AS [System Cache (MB)],
       system_memory_state_desc AS [System Memory State]
FROM sys.dm_os_sys_memory WITH (NOLOCK) OPTION (RECOMPILE);
```

5.4. Query 4: Input/output statistics by file for the current database

To see input /output (I/O) statistics by file for the current database, run the following query against your OpenText Application Security database. This helps you better characterize your workload from an I/O perspective for the database.

```
SELECT DB_NAME(DB_ID()) AS [Database Name], df.name AS [Logical Name], vfs.[file_id],
df.physical_name AS [Physical Name], vfs.num_of_reads, vfs.num_of_writes, vfs.io_stall_read_ms,
vfs.io_stall_write_ms,
CAST(100. * vfs.io_stall_read_ms/(vfs.io_stall_read_ms + vfs.io_stall_write_ms) AS DECIMAL(10,1)) AS [IO
Stall Reads Pct],
CAST(100. * vfs.io_stall_write_ms/(vfs.io_stall_write_ms + vfs.io_stall_read_ms) AS DECIMAL(10,1)) AS
[IO Stall Writes Pct],
(vfs.num_of_reads + vfs.num_of_writes) AS [Writes + Reads],
CAST(vfs.num_of_bytes_read/1048576.0 AS DECIMAL(10, 2)) AS [MB Read],
CAST(vfs.num_of_bytes_written/1048576.0 AS DECIMAL(10, 2)) AS [MB Written],
CAST(100. * vfs.num_of_reads/(vfs.num_of_reads + vfs.num_of_writes) AS DECIMAL(10,1)) AS [# Reads Pct],
CAST(100. * vfs.num_of_writes/(vfs.num_of_reads + vfs.num_of_writes) AS DECIMAL(10,1)) AS [# Write Pct],
CAST(100. * vfs.num_of_bytes_read/(vfs.num_of_bytes_read + vfs.num_of_bytes_written) AS DECIMAL(10,1))
AS [Read Bytes Pct],
CAST(100. * vfs.num_of_bytes_written/(vfs.num_of_bytes_read + vfs.num_of_bytes_written) AS
DECIMAL(10,1)) AS [Written Bytes Pct]
FROM sys.dm_io_virtual_file_stats(DB_ID(), NULL) AS vfs
INNER JOIN sys.database_files AS df WITH (NOLOCK)
ON vfs.[file_id]= df.[file_id] OPTION (RECOMPILE);
```

Example output

Example output of input and output statistics by file

5.5. Query 5: Volume information for all logical unit numbers with database files on the current instance

To see details about the logical unit numbers (LUNS) that have database files on the current instance, run the following query against the main database:

```
SELECT DISTINCT vs.volume_mount_point, vs.file_system_type,
vs.logical_volume_name, CONVERT(DECIMAL(18,2),vs.total_bytes/1073741824.0) AS [Total Size (GB)],
CONVERT(DECIMAL(18,2),vs.available_bytes/1073741824.0) AS [Available Size (GB)],
CAST(CAST(vs.available_bytes AS FLOAT)/ CAST(vs.total_bytes AS FLOAT) AS DECIMAL(18,2)) * 100 AS [Space
Free %]
FROM sys.master_files AS f WITH (NOLOCK)
CROSS APPLY sys.dm_os_volume_stats(f.database_id, f.[file_id]) AS vs OPTION (RECOMPILE);
```

This enables you to see the total space and free space on the LUNs where you have database files.

Example output

example output from query 5

5.6. Query 6: Volume data for all LUNS that have database files on the current instance

To see volume information for all LUNS that have database files on the current instance, run the following query against the main database:

```
CREATE TABLE #IOWarningResults(LogDate datetime, ProcessInfo sysname, LogText nvarchar(1000));
       INSERT INTO #IOWarningResults
       EXEC xp_readerrorlog 0, 1, N'taking longer than 15 seconds';
       INSERT INTO #IOWarningResults
       EXEC xp_readerrorlog 1, 1, N'taking longer than 15 seconds';
       INSERT INTO #IOWarningResults
       EXEC xp_readerrorlog 2, 1, N'taking longer than 15 seconds';
       INSERT INTO #IOWarningResults
       EXEC xp_readerrorlog 3, 1, N'taking longer than 15 seconds';
       INSERT INTO #IOWarningResults
       EXEC xp_readerrorlog 4, 1, N'taking longer than 15 seconds';
SELECT LogDate, ProcessInfo, LogText
FROM #IOWarningResults
ORDER BY LogDate DESC;
DROP TABLE #IOWarningResults;
```

Finding 15-second I/O warnings in the SQL Server Error Log is evidence of poor I/O performance (which might have any number of different causes).



Note

No data should be returned from this query, which is a good thing.



Note

Depending on the number of records returned, and the frequency with which they are returned, consult with your storage team to review the errors.

5.7. Query 7: Drive-level latency information

To view drive-level latency information, run the following query against the main database:

```
SELECT [Drive],
       CASE
              WHEN num_of_reads = 0 THEN 0
              ELSE (io_stall_read_ms/num_of_reads)
       END AS [Read Latency],
       CASE
              WHEN io_stall_write_ms = 0 THEN 0
              ELSE (io_stall_write_ms/num_of_writes)
       END AS [Write Latency],
       CASE
              WHEN (num_of_reads = 0 AND num_of_writes = 0) THEN 0
              ELSE (io_stall/(num_of_reads + num_of_writes))
       END AS [Overall Latency],
       CASE
              WHEN num_of_reads = 0 THEN 0
              ELSE (num_of_bytes_read/num_of_reads)
       END AS [Avg Bytes/Read],
       CASE
              WHEN io_stall_write_ms = 0 THEN 0
              ELSE (num_of_bytes_written/num_of_writes)
       END AS [Avg Bytes/Write],
       CASE
              WHEN (num_of_reads = 0 AND num_of_writes = 0) THEN 0
              ELSE ((num_of_bytes_read + num_of_bytes_written)/(num_of_reads + num_of_writes))
       END AS [Avg Bytes/Transfer]
FROM (SELECT LEFT(UPPER(mf.physical_name), 2) AS Drive, SUM(num_of_reads) AS num_of_reads,
                SUM(io_stall_read_ms) AS io_stall_read_ms, SUM(num_of_writes) AS num_of_writes,
                SUM(io_stall_write_ms) AS io_stall_write_ms, SUM(num_of_bytes_read) AS
num_of_bytes_read,
                SUM(num_of_bytes_written) AS num_of_bytes_written, SUM(io_stall) AS io_stall
      FROM sys.dm_io_virtual_file_stats(NULL, NULL) AS vfs
      INNER JOIN sys.master_files AS mf WITH (NOLOCK)
      ON vfs.database_id = mf.database_id AND vfs.file_id = mf.file_id
      GROUP BY LEFT(UPPER(mf.physical_name), 2)) AS tab
ORDER BY [Overall Latency] OPTION (RECOMPILE);
```

Example output

The following table shows you the drive-level latency for reads and writes, in milliseconds.



Reference table

Milliseconds	Indication
< 1	Excellent
< 5	Very good
5–10	Good
10–20	Marginally acceptable
20–100	Bad
100–500	Very bad
>500	Extremely bad



Note

Latencies above 20 to 25 milliseconds usually indicate a problem exists with the storage system that hosts your OpenText Application Security database. In such cases, contact your storage team to discuss reducing the Read/Write latency.

5.8. Query 8: Index fragmentation

To check for index fragmentation in your OpenText Application Security database, run the following query against your database:

```
SELECT OBJECT_NAME(OBJECT_ID), index_id,index_type_desc,index_level,
avg_fragmentation_in_percent,avg_page_space_used_in_percent,page_count
FROM sys.dm_db_index_physical_stats
(DB_ID(N'SSC'), NULL, NULL, NULL , 'SAMPLED')
ORDER BY avg_fragmentation_in_percent DESC
```

Example output



This output Indicates the following fragmentation levels in the OpenText Application Security database:



If you have a SQL job configured to run a maintenance plan to rebuild or reorganize the indexes, check out IndexOptimize, SQL Server Maintenance Solution's stored procedure for rebuilding and reorganizing indexes and updating statistics. For details, see SQL Server Maintenance Solution.

You can also use Microsoft's AdaptiveIndexDefrag to perform an Intelligent defrag on one or more indexes, and a required statistics update. For details on what AdaptiveIndexDefrag does and how to use it, see Adaptive Index Defrag.

5.9. Query 9: SQL Version

To determine the SQL version of your database, run the following query:

```
Select @@Version
```

5.10. Enable Query Store for the Application Security database

If you want to collect data for a minimum of 24 hours. On the SQL Server **Database Properties** page, set the properties with the values shown in the following table.

Property	Setting
Operation Mode	Read write
Statistics Collection Interval	5 Minutes You can collect statistics every 1 minute.
Max Size (MB)	The default value is 1000. Add a few zeros to this value. You can purge the data later.
Wait Statistics Capture Mode	On

The Query Store includes several queries.



> **Note**
>
> For Azure Synapse Analytics, Query Store views are available under **System Views** in the database portion of the Object Explorer pane.

5.11. SQL Scripts: First Responder Kit

Brent Ozar (Microsoft Certified Master, SQL Server consultant, and trainer) offers a free First Responder Kit to help you analyze and tune your SQL database. It includes the scripts described in the following table.

Script	Purpose
sp_Blitz	If you acquire a database and you are uncertain about its health, you can run this script to perform a database health assessment that quickly flags common issues. For each issue uncovered, the script provides a link to a web page with more in-depth advice. For details, see the sp_Blitz® Documentation. To see a video demo on how to use the script, see the sp_Blitz® – Free SQL Server Health Check Script webpage.
sp_BlitzFirst	This script helps troubleshoot slow SQL Servers by quickly: • Blocking long-running queries • Determining whether a backup, database console command (DBCC), or index maintenance job was running • Locating any SQL Server bottlenecks • Checking Perfmon counters for CPU use, slow drive response times, or low Page Life Expectancy To view a video on how to use the script, see https://www.brentozar.com/askbrent.
sp_BlitzCache	Use this script to determine which queries are causing the biggest performance problems and what you can do about them. For details about the script and to view a video on how to use it, see sp_BlitzCache®: Find Your Worst-Performing Queries.
sp_BlitzIndex	Use this script to conduct a sanity check and report on your database and diagnose your indexes major disorders. For each detected disorder, a URL is provided that explains what to look for and how to handle the issue. The script also enables you to see both the "missing" and existing indexes for a table in a single view. For more details, and to see a video on how to use the script, see sp_BlitzIndex® – SQL Server's Index Sanity Test.
sp_BlitzLock	Use this script to analyze deadlocks and determine what queries and tables you need to change. For detailed information about the script, see Introducing sp_BlitzLock: For Troubleshooting SQL Server Deadlocks.

© Copyright 2026 Open Text

For more info, visit https://docs.microfocus.com