SQL Server Configuraton

Version 8.1.6, December 2019

Contents

Introduction

Cb Protection uses a single SQL database named “das”. This database will be created on the SQL server once you install the Cb Protection Server.

This document describes some of SQL best practices as well as pre- and post-installation tasks that are necessary to ensure proper database operation.

Most of the database queries described in this document require sysadmin privileges on the SQL Server.

Important: This document is a supplement to the *Cb Protection Operating Environment Requirements* document for your release, called the “OER” through most of this document. Information in the OER is necessary for successful completion of the tasks in this document, so you should have both documents with you when performing the tasks described here.

General Performance Notes

Cb Protection Server relies on a SQL Server database for storage of all file, computer, and other data it collects. Therefore, the performance of Cb Protection Server is strongly influenced by the performance of the underlying SQL Server.

Here is an outline of the basic guidelines that will help SQL Server perform well. Most of these are explained in the detail throughout the remainder of this document.

1. The SQL Server must be exclusively dedicated to the Cb Protection Server. Do not put any other database on this SQL Server instance.
2. Before the Cb Protection Server installation, the performance of the database hardware should be validated using the methods outlined in this document.
3. After the Cb Protection Server is successfully installed, you will need to manually adjust some of the database and SQL server properties for optimal performance, as described in this document.
4. The SQL Server must be on the same physical machine as the Cb Protection Server. In the exceptional case when this is not possible, special care must be taken to ensure good network performance between the two machines.
5. SQL Server storage must be configured and must perform correctly.
6. The SQL Server should have enough memory, and the memory cap should be configured correctly.
7. Cb Protection Server will do all of the required database maintenance work on a regular schedule. Avoid any other maintenance tasks on the Cb Protection database.
8. Use the recommended SQL Server backup strategy, as outlined in the separate document, “*Cb Protection High Availability and Disaster Recovery Guide”.*

Cb Protection Server One-tier Architecture

In order to simplify configuration and ensure fast connectivity between the Cb Protection server and SQL server, the Cb Protection “Operating Environment Requirements” (OER) specifies that both products must be installed on the same machine.

When deploying the Cb Protection in this manner, SQL Server should be configured to communicate with the Cb Protection Server using *shared memory* rather than TCP/IP.

Cb Protection Server Two-tier Architecture

Sometimes, however, a one-tier deployment of Cb Protection is not possible because it conflicts with company policies mandating that all databases must be located in a data center and managed by a separate team.

If the Cb Protection Server and SQL Server reside on separate machines, the following precautions must be taken:

Choosing Proper Hardware for the SQL Server

All OER requirements for a given deployment size must be followed when choosing hardware for the separate SQL server. There are only two exceptions to this rule:

- Since the Cb Protection Server will now reside on a separate machine, the CPU requirements for the SQL Server machine can be reduced by 2 cores for all deployments over 10,000 agents.
- The SQL Server machine can use Windows Server 2019, Windows Server 2016, Windows Server 2012 R2, or Windows Server 2008 R2 as its operating system.

Important: Regardless of whether you use a one-tier or two-tier architecture, the SQL Server machine and its database storage still must be dedicated to Cb Protection.

Choosing Proper Hardware for Cb Protection Server

The Cb Protection Server should be built using following minimum requirements:

Operating System	Windows Server 2008 R2
RAM	16 GB
CPU	Intel Xeon or similar, with 6 physical cores

Network Configuration

The Cb Protection Server machine should follow OER requirements for opening network ports.

The Cb Protection Server and SQL Server must be connected by a fast network link meeting the following requirements:

Network speed	>800 Mb/s (megabits per second)
Network latency	<1ms

You can best achieve these numbers by using unswitched/unrouted connection (e.g. cross-link cable).

Network Latency Impact

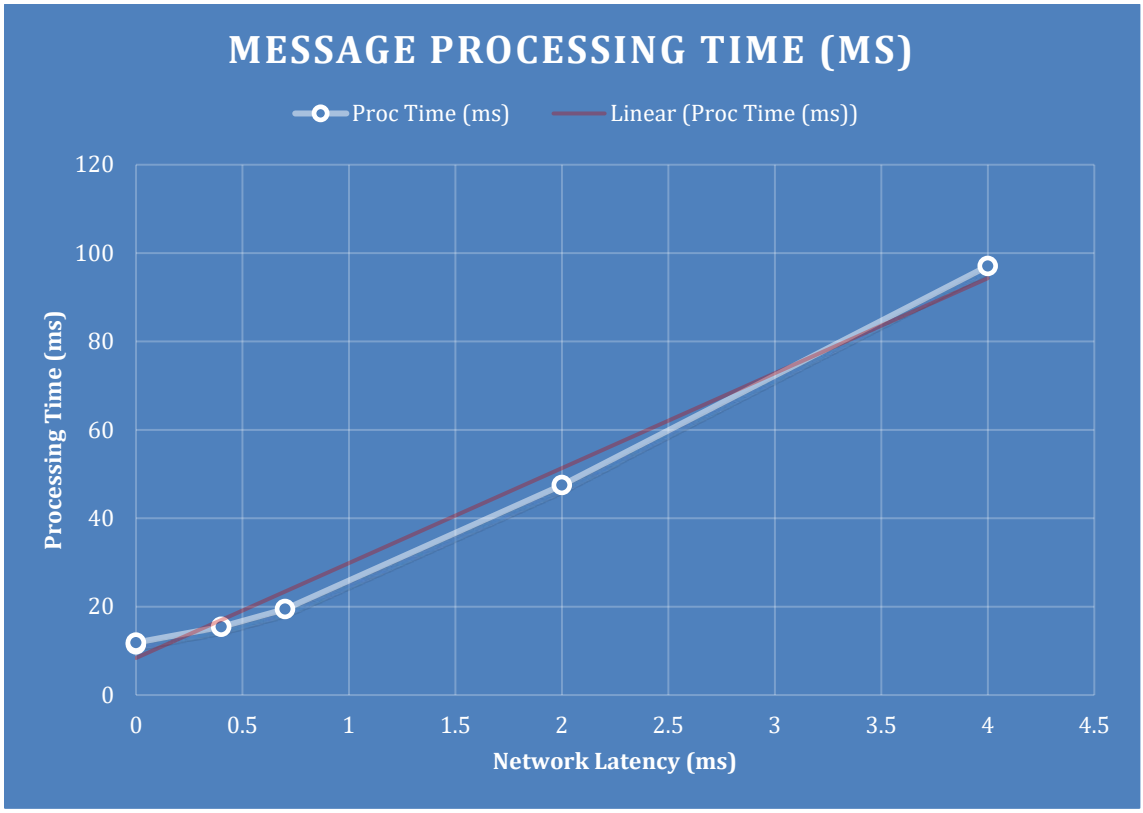
Network latency can pose a significant impact on the performance of the Cb Protection.

The Cb Protection Server processes agent messages at a very high rate — sometimes exceeding 20 million per day. This directly translates into a high rate of SQL server commands (1000 per second or more). In the two-tier scenario, each of these commands needs to be sent over the network connection.

For example, if the number of commands sent from a single product thread achieves a rate of 1000 per second, each command has only 1 ms to complete. Any additional network latency

therefore creates a large impact (0.5 ms latency in each direction doubles the time for each command to complete).

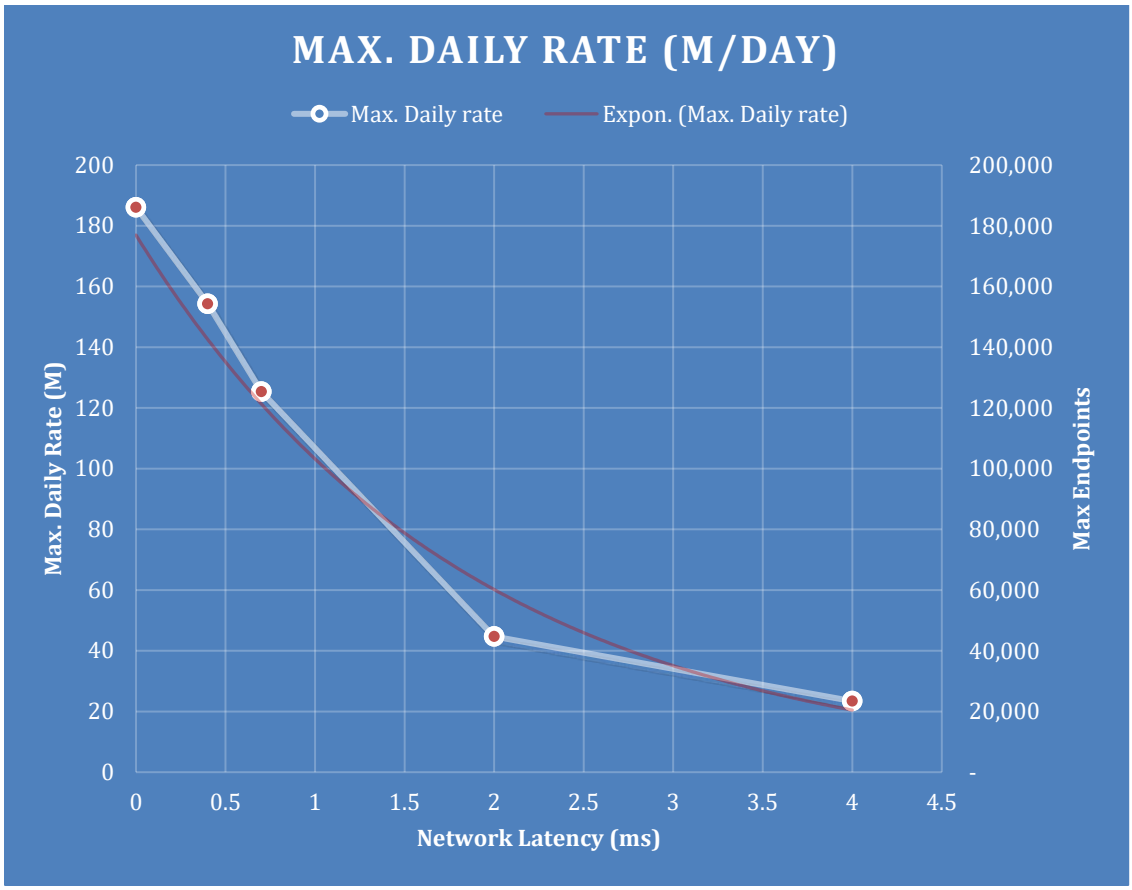
Internal Carbon Black lab measurements confirm this expectation, where it can be seen that the dependency of maximum server message processing throughput on network latency can be well described with linear regression:



Best results (left side of the graph) were achieved with a 1-tier deployment using a shared memory connection between Cb Protection and SQL servers.

Results on the right are retrieved using 2-tier with increasing network latency.

This also reflects in measurements of maximum server throughput (exponential trendline). As can be seen here, adding 2ms to the network throughput (1 ms in each direction) will reduce server throughput from 190 to 45 (more than 75% reduction).



Note: *This graph represents the maximum daily message processing rate achieved with a server sized for 20,000 endpoints with varying network latency. Other factors, like SQL or storage performance, can additionally limit this rate.*

Depending on type of environment and endpoint update schedule, an average agent can produce between 500 and 2,000 messages per day. Therefore, the daily rate in Millions roughly correspond to the number of supported agents in thousands.

SQL Permissions

Please make sure that following permissions are granted to the “Cb Protection” user:

1. “Cb Protection” user has to be set as **db_owner** on Das database.
2. Following server-level permissions should be granted in order to allow diagnostic collection:

Permission	Required	Reason
VIEW SERVER STATE	Yes	Allows collection of Cb Protection performance statistics
VIEW ANY DEFINITION	Yes	Allows collection of Cb Protection performance statistics
ALTER TRACE	Yes	Allows collection of on-demand SQL trace for performance diagnostics
ALTER SERVER STATE	No (Recommended)	Allows server to reset performance counters on daily basis, and provides better performance diagnostics

Note: Even in shared environments, Cb Protection Server will require sysadmin permission during the time of server installation. In that case, sysadmin permission can be taken away after the install has completed.

SQL Server Memory Configuration

The OER specifies the minimum memory requirements for a desired deployment size.

SQL Server allows you to customize the memory limit, to make sure the server will never consume all system memory.

Consult the following Microsoft documentation to set the maximum amount of memory:

http://technet.microsoft.com/en-us/library/ms191144(v=sql.105).aspx

The following table outlines how much memory should be made available to SQL Server, for different deployments:

Deployment	Required total RAM on the server	Memory Limit to Set for SQL Server
101 – 250 endpoints	12 GB	8 GB = 8192 MB
251 – 2,000 endpoints	16 GB	12 GB = 12288 MB
2,001 – 5,000 endpoints	32 GB	27 GB = 27648 MB
5,001 – 20,000 endpoints	48 GB	38 GB = 38912 MB
20,001 – 30,000 endpoints (SQL Standard)	128 GB	118 GB = 120832 MB
10,001 – 40,000 endpoints (SQL Enterprise)	64 GB	54 GB = 55296 MB
Over 40,000 endpoints (SQL Enterprise)	96 GB	86 GB = 88604 MB

Note: Deployments smaller than 100 endpoints require SQLExpress only, which has a built-in maximum memory cap of 1 GB.

In order to restrict SQL memory usage, you can run following query from SQL management Studio (replacing the highlighted number with number of megabytes from the table above):

```
EXEC sys.sp_configure N'show advanced options', N'1'  RECONFIGURE WITH OVERRIDE
GO
EXEC sys.sp_configure N'max server memory (MB)', 53248
GO
RECONFIGURE WITH OVERRIDE
GO
EXEC sys.sp_configure N'show advanced options', N'0'  RECONFIGURE WITH OVERRIDE
GO
```

Cb Protection Server Database Storage

The main bottleneck for the SQL Server performance is its database storage. In order to ensure fast storage performance, these steps should be followed:

1. Storage should meet OER requirements
2. SQL Server physical storage must be dedicated to the Cb Protection Server database. Do not put other applications, OS files or data on this storage.

3. Exclude the SQL Server data directories from your AV scanners or other tools that might impact the disk performance.
4. Use the Bit9SQLIO tool to validate that the performance of underlying storage satisfies requirements outlined in OER.

Database Files

The Cb Protection Server database consists of five different files. Additionally, SQL Server keeps two temporary files for its operations. The following table lists all of the relevant database files:

Filegroup Name	File name	Description	Type
PRIMARY	das.mdf	Stores data that is not related to inventory or events	Data
SECONDARY	das_events.ndf	Stores event data	Data
ABINST	das_abinst.ndf	Stores inventory data	Data
ABTEMP	das_abtemp.ndf	Stores inventory indexes	Index
Log file	das_log.ldf	Stores transaction log	Log
Temp SQL data file	tempdb.mdf	Stores temporary SQL data	Temp
Temp SQL log file	tempdb.ldf	Stores temporary SQL data	Temp

Note: Same color coding for different file types is used throughout this chapter.

As mentioned in the OER, the Cb Protection database should be stored on direct attached storage (DAS). For larger deployments (over 10,000 endpoints), additional flash SSD storage should be used.

There are three ways to split database files, depending on desired performance, size and file type:

Layout	Endpoints	Data	Index	Log	Temp	AEFW[2]
Standard performance	<10,000	DAS	DAS	DAS	DAS	N/A
	>=10,000	DAS	Flash	DAS	DAS	13 GB
High performance[4]	Any[1]	Flash	Flash	DAS[3]	Flash	20 GB

*[1] The High-performance configuration is not **required** for any deployment size. However, it will provide the fastest response for Cb Protection Server console activity. Note, however, that it will also increase total hardware cost since flash storage is still more expensive than hard disks.*
[2] AEFW represents Annual disk writes to the flash storage in bytes for single endpoint. It is used in the flash storage lifetime calculation in the next section.
[3]Transaction Log Files should never be put on the flash storage. DAS storage usually has equal or better sequential write performance than flash, so it is not required. Also, given that flash storage usually has a limit on lifetime number of writes, it is better to store log files on less expensive DAS drives.
[4]"High performance" mode will provide up to 30% higher performance for most server operations

SSD and Flash Storage Lifetime

SSD and Flash storage usually has manufacturer-defined lifetime defined by total amount of writes.

To compare this number with expected number of writes for the Cb Protection Server database, the following calculation can be used:

Estimated lifetime disk writes = **AEFW** × **Agents** × **Years**

AEFW: number depends on database layout and can be obtained from table in the previous section.

Agents: total number of agents installed from this Cb Protection server. Note that agents are busiest during initialization, and less during stead-state. Therefore, use total number of agents you plan to install for this number, even if some agents will be uninstalled at later time.

For example, if your environment will have **100K** endpoints installed, and you want to make sure that lifetime of the card will exceed **5** years, flash card will accumulate 13 GB $\times$ 100,000 $\times$ 5 $\approx$ **6.5 PB** (petabytes) of writes in its lifetime.

Note that modern Flash SSD cards typically have more than 10 PB (petabytes). Therefore, life-time of SSD cards should not be an issue for Cb Protection Server.

Database Size Calculator

The following table helps you determine the total storage size based on the number of endpoints:

File Type	Size per 1,000 endpoints For deployments under 5,000 endpoints	Size per 1,000 endpoints For deployments between 5,000 and 10,000 endpoints	Size per 1,000 endpoints For deployments over 10,000 endpoints[2]
Data	90 GB	55 GB	20 GB
Index	50 GB	25 GB	20 GB[1]
Log	50 GB	25 GB	10 GB
Temp	35 GB	15 GB	5 GB
Total	**225 GB**	**120 GB**	**55 GB**

[1] Flash PCIe storage is required for Index files for deployments over 10,000 endpoints
[2] SQL database deployments over 10K endpoints requires SQL Server Enterprise. The figures in the table assume use of database compression.

Database size is calculated differently for different deployment sizes for two major reasons:

1. SQL Server Standard edition does not provide SQL database compression.
2. Some SQL tables grow in a non-linear way.

You can pro-rate this data for desired table size.

Here are few examples of how you would use this data to determine your database size:

Example 1. Deploying Cb Protection server for 8,000 endpoints

a. According to the OER, a single DAS partition is needed for 8,000 endpoints.
b. In the table, the guidelines for 8000 end points are in the second column (5,000 – 10,000 endpoints).
c. Based on the table guidelines, the total storage needed is 120 GB × 8 = **960 GB**

Example 2. Deploying Cb Protection server for 30,000 endpoints

We will calculate the storage needed for 30,000 endpoints by referring to the table titled *Cb Protection Server Architecture by Endpoint Count* from the OER and to the database size calculator above.

a. According to the OER table, one DAS partition and a second partition on PCIe flash storage are needed for 30,000 endpoints.
b. We must prorate the values from the OER table. Dividing 30,000 endpoints by 40,000 gives a factor of **0.75**. The OER table indicates that 2 TB would be needed for 40,000 endpoints. Thus, **1.5 TB** in total would be needed for a 30,000 endpoint deployment.
c. In the calculator table above, the third column (deployments over 10,000 endpoints) provides the guidelines for 30,000 endpoints. To get the values we need for the index file for 30,000 endpoints, we calculate 20 GB × (30,000 endpoints / 1,000 endpoints) giving us **600 GB** for index storage.
d. Since the rest of the database (Data+Log+Temp) needs to go to the DAS storage, we will calculate the prorated size required as we want 1. 5 TB – 600 GB = 1,536 GB – 600 GB = **936 GB** for the remaining storage.

Example 3. Deploying Cb Protection server for 100,000 endpoints with maximum performance

We will calculate the storage needed by referring to the table titled *Cb Protection Server Architecture by Endpoint Count* from the OER and to the database size calculator above.

a. According to the OER table, one DAS partition and second partition on PCIe flash storage are needed for 80,000 endpoints. For maximum performance, however, the OER also recommends putting the entire database, except for the Log files, on the PCI storage.
b. In the database size calculator table, the third column provides the guidelines for 100,000 endpoints. The log files, which need to go into DAS storage, will have a size of 10 GB × (100,000 / 1,000) = 1,000 GB = 0.98 TB
c. The remaining files (data, temp, and index) must go on flash storage. We prorate the values in the OER table by multiplying by 100,000 / 80,000 = **1.2**.
d. The OER table indicates that the total database size is 4 TB. Prorating gives 4 TB × 1.2 = 4.8 TB.
e. The storage required on flash storage is thus 4.8 TB – 0.98 TB = **3.8 TB**.

Moving Database Files after Installing Cb Protection Server

Installation of the Cb Protection Server installs all of the above-mentioned database files to the default location specified in the SQL Server. Once a brand new installation of Cb Protection Server is completed, you should move these files to the new locations, based on storage configuration and guidelines in previous section.

Before moving database files, you must stop the following Cb Protection services:

- Cb Protection Server
- Cb Protection Reporter

You can move files through the SQL Management Studio, using the query window.

To move the Cb Protection database files:

1. Locate current database files using following query

```
SELECT Db_Name(database_id) as database_name, name, physical_name FROM sys.master_files
WHERE Db_Name(database_id) IN ('das', 'tempdb')
```

This will return the location of all files. For example:

database_name	name	physical_name
tempdb	tempdev	E:\Microsoft SQL Server\MSSQL10_50.MSSQLSERVER\MSSQL\DATA\tempdb.mdf
tempdb	templog	E:\Microsoft SQL Server\MSSQL10_50.MSSQLSERVER\MSSQL\DATA\templog.ldf
Das	das	E:\Microsoft SQL Server\MSSQL10_50.MSSQLSERVER\MSSQL\DATA\das.mdf
Das	das_log	E:\Microsoft SQL Server\MSSQL10_50.MSSQLSERVER\MSSQL\DATA\Das_log.LDF
Das	das_events	E:\Microsoft SQL Server\MSSQL10_50.MSSQLSERVER\MSSQL\DATA\Das_events.ndf
Das	das_abinst	E:\Microsoft SQL Server\MSSQL10_50.MSSQLSERVER\MSSQL\DATA\Das_abinst.ndf
Das	das_abtemp	E:\Microsoft SQL Server\MSSQL10_50.MSSQLSERVER\MSSQL\DATA\das_abtemp.ndf

2. Next, use the following command to move each file:

```
ALTER DATABASE <database_name> MODIFY FILE (NAME = <file_name>, FILENAME = <location>);
```

For example, suppose that the F: drive is PCIe flash storage and the D: drive is DAS storage. We want to move all index file to F:\SQL drive and all other files to D:\SQL. This is what we would do:

```
ALTER DATABASE Das MODIFY FILE (NAME = das, FILENAME = 'D:\sql\das.mdf');
ALTER DATABASE Das MODIFY FILE (NAME = das_log, FILENAME = 'D:\sql\das_log.ldf');
ALTER DATABASE Das MODIFY FILE (NAME = das_abinst, FILENAME = 'D:\sql\das_abinst.ndf');
ALTER DATABASE Das MODIFY FILE (NAME = das_events, FILENAME = 'D:\sql\das_events.ndf');
ALTER DATABASE Das MODIFY FILE (NAME = das_abtemp, FILENAME = 'F:\sql\das_abtemp.ndf');
ALTER DATABASE tempdb MODIFY FILE (NAME = tempdev, FILENAME = 'D:\sql\trempdb.mdf');
ALTER DATABASE templog MODIFY FILE (NAME = tempdev, FILENAME = 'D:\sql\tremplog.ldf');
```

3. Once all database files are moved in SQL, stop the SQL Server service.

4. From Windows Explorer, move all the database files to their new locations. There will be 5 files from the Cb Protection “das” database, and also 2 files from the System “tempdb”.

5. Start the SQL Server service. Make sure that the database is started successfully by opening it from SQL Management Studio.

6. Start both Cb Protection services

Cb Protection Server and Storage Area Networks

Even though the *Cb Protection Operating Environment Requirements* document doesn’t recommend running Cb Protection Server on SAN (Storage Area Networks), there are situations where this cannot be avoided. This document provides guidelines and requirements for a Cb Protection Server setup that uses a SAN for database storage.

Here are some common reasons to use a SAN with Cb Protection Server:

- *“Because, per corporate standards, we need to run Cb Protection Server in a SQL cluster”*
- This is a valid reason for using a SQL cluster. Note that SQL Server Always On availability groups, available in SQL Server 2012, don’t have this limitation.
- *“Because, per corporate standards, we want to run Cb Protection Server in a virtualized environment, and use SAN as storage.”*
- This is a valid reason since it is difficult to provision VM with DAS, especially if VM images will be moved across different hardware.

There are downsides to using a SAN with Cb Protection Server:

- The SAN usage model usually implies that storage will be shared with other databases and applications and it is unlikely it is going to meet the Cb Protection Server performance requirements.
- SAN storage that will meet the performance requirements for Carbon Black is very expensive. Carbon Black recommends using DAS in combination with Flash storage. This combination provides better and more reliable results than SAN for a much lower price.

Using a SAN with Cb Protection Server

Requirements

1. When a SAN is used for database storage, Cb Protection Server hardware still must meet the other physical requirements described in the Cb Protection Operating Environment Requirements document.
2. For the best performance, Cb Protection Server and SQL Server should both be deployed on a single machine (1-tier architecture).
3. SAN storage must be qualified using the performance validation tools provided by Carbon Black (described in this document), and must meet the minimum performance requirements for a given deployment size.
4. The storage size must meet minimum requirements for a given deployment size.
5. The SAN must be configured to have LUNs dedicated to Cb Protection Server (e.g., through “zoning”).

Best Practices when Using a SAN with Cb Protection Server

1. Configure the HBA controller for a high queue depth (24 or more).
2. Use a fast Fibre-channel connection (at least 8 GB/s).
3. When using a SAN with hard disks, use RAID 1+0 if available; it will give the best performance.
4. Use a stripe size of either 64Kb or 256Kb.
5. To meet the performance requirements for Cb Protection, a SSD might need to be used inside of a SAN. If so, the SSD cache must be dedicated and large enough to cover approx. 30% of the database size.

Validating Storage Performance

There are two approaches to validation of storage performance with Cb Protection Server.

1. **I/O Performance test** – Depending on whether you are using direct-attached storage or a SAN, we recommend different approaches to testing I/O performance:
    a. **Direct-attached storage: Bit9SQLIO** - This is a lightweight, 30-minute test that uses the SQLIO tool to measure storage performance.
    b. **SAN storage**: **diskspd** – If you are using a SAN, you should instead do two 20-minute runs of the **diskspd** utility, downloadable from Microsoft here: https://gallery.technet.microsoft.com/DiskSpd-a-robust-storage-6cd2f223 Please make sure you have 100GB of free space on the disk you are testing.
    Run the following command lines on those disks (each will run for 20 minutes) and send the results back to sales engineering:

```
diskspd -w100 -t1 -b40K -d1200 -o1 -fs -Su -L -c100G testfile.dat
diskspd -w100 -t1 -b8K -d1200 -o32 -r -fr -Sh -L -c100G testfile.dat
```

2. **SQL Performance Test** – This is a more comprehensive (1-8 hour) test that uses the actual SQL Server and a backup of a large database to determine the performance of

storage. This test should be performed when using a non-standard deployment, such as SAN.

Both tests must meet the required metrics based on desired deployment size, as noted for each test.

Bit9SQLIO Test

The Bit9SQLIO test measures storage performance using access patterns that closely resemble those for SQL Server when used by the Cb Protection Server. The following metrics are measured:

1. **Sequential Writes**: Measured as throughput in MB/s of sequential writes with block size of 40 KB, and no outstanding requests (raw IOPS using a single level of parallelism).
2. **Random Reads**: Measured as throughput in MB/s of random reads with block size of 8 KB, and 4 outstanding requests.
3. **Random Writes**: Measured as throughput in MB/s of random writes with block size of 8 KB, and 32 outstanding requests.

Required Performance Metrics

The following table describes the metrics that the storage must meet in order to support different numbers of endpoints. For storage to meet Cb Protection requirements, its metrics must be **equal or above** the number in the table in each category.

Max Endpoints	Bit9SQLIO Metrics (MB/s)		
	Sequential Writes	Random Reads	Random Writes
0 – 500	5	5	5
501 – 1,500	12	10	25
1,500 – 2,000	12	10	40
2,000 – 5,000	25	15	40
5,001 – 10,000	50	25	60
10,001 - 40,000	250	30	60
40,001 - 80,000	250	60	120
80,001 - 160,000	250	120	240

Test Prerequisites

1. The System needs 100 GB of free space on the drives tested. The special –f switch described below can reduce this requirement, but note that using smaller files will also reduce the accuracy of results because of disk caching.
2. Test requires .Net Framework 3.5 or newer to be installed on the system. Note that many Windows OS versions already come with version 3.5 or later pre-installed. If you are running this test from an older OS (older than Windows 7 or Windows Server 2008 R2), you can download this installer from the official Microsoft web site: http://www.microsoft.com/en-us/download/details.aspx?id=30653

Running Bit9SQLIO

The Bit9SQLIO tool is run at a command prompt and has a simple syntax, as shown in the following examples:

```
>Bit9SqlIO f:
```
This runs Bit9SqlIO on drive f:

```
>Bit9SqlIO f:,g:
```
This runs Bit9SqlIO on drives f: and g:

Note: Normally, this test requires at least 100 GB free space on the disk(s) being tested. However, there is an additional parameter (**–f**) to change the size of the test file for environments when this amount of space is not available.

```
>Bit9SqlIO f: -f 50
```

In this example, temporary testing file is reduced to 50 GB. This can be done only if total database size will not be larger than 50 GB.

The typical duration of the entire test will be 20 minutes per disk.

Example Output for Bit9SqlIO:

```
C:\Bit9SqlIO>Bit9SqlIO.exe e:
* 10:10:48 AM - Running Bit9SqlIO on e: using 1 threads.
* Approximate duration of the test will be 20 minutes

*** Result summary for drive h: ***
Operation         Ous  MB/s   IOPS  Lat Endpoints
----------------- --- ----- ------ ---- ---------
Sequential Writes   1   225   5755    0    102123
Random Reads        4     6    803    0    126412
Random Writes      50    47   6116    1    264174
***
* 10:22:05 AM – Completed
```

The important output metrics are under the column MB/s in the results, and should be correlated with the requirements in the table above, taking into account the desired endpoint count.

Logging of Bit9SqlIO Results:
In addition to the simplified tabular output in the command window, Bit9SqlIO creates a detailed log file in the same directory as its executable. The log will contain additional analysis information. It is additive, and will add to existing log file every time the tool is run.

The log should be sent to engineering upon request.

SQL Performance Test

Description

The full SQL Performance test is a storage qualification method that should be used for testing SAN storage for deployments with 20K or more endpoints. It can also be used for smaller deployments when Bit9SQLIO doesn't provide adequate results, since this test is a stronger validation of the storage.

This test requires that a relatively large amount of SAN storage be available (approximately 350 GB), and it takes several hours to complete.

Running SQL Performance Test

The SQL Performance test consists of several files:

1. Backup.bak – A database backup (~30 GB)
2. SQLPerformanceTest.bat - A batch script that performs the test
3. Two SQL scripts (restore.sql and run.sql) used by the batch script

Copy all four files to the directory on your SQL server machine. It can be on one of the disks you will be testing.

SQL Performance Test Syntax and Parameters:

```
>SQLPerformanceTest.bat <server> <backup> <data> [<log> <index> <temp>]
```

- **server**: Name of local SQL Server instance or "." for default instance
- **backup**: Location of database backup. Use"." for default backup (use "." unless backup was manually moved to a different location than ".\backup.bak")
- **data**: Location of SQL data files (e.g., "f:\data"). This includes PRIMARY, SECONDARY and ABINST filegroups.
- **log**: (Optional) location of SQL log files (e.g., "i:\log") - defaults to the value for <data> if not specified.
- **index**: (Optional) location of SQL index files (e.g. "g:\index") - defaults to the value of <data> if not specified. This includes ABTEMP filegroup.
- **temp**: (Optional) location of SQL temp files (e.g., "h:\temp") - defaults to the value of <data> if not specified.

Notes:

1. *Data, Index, Temp and Log must point to the same location as planned for Cb Protection database.*
2. *The script assumes that the SQL Server service name will be "MSSQL$<instance>". If the name of the service is different, the script will have to be modified manually.*
3. *The script assumes that the console user will have **sysadmin** access to the SQL Server instance through Windows authentication. You might need to use CMD as "Run as…" in order to ensure this.*

Example:

```
>SQLPerformanceTest.bat . . "F:\SQL"
```

This will perform the test on default SQL instances, and put all database files into directory f:\SQL.

SQL Performance Text Script Output

The script will create a log file in the location where it was run. This log file should be sent to engineering upon request.

The console output will look like this:

```
Metric name                          Duration
-------------------------------- -----------
Backlog                                 3563
Counting                                1443
Pruning                                  750
```

Metrics

The following table describes the metrics that storage must meet to support different numbers of endpoints. For storage to qualify for Cb Protection, each of the metrics must be **equal or below** the number in the table. Note that test itself might require more disk space than Cb Protection database for small deployment size.

Endpoints	SQL Performance Script Metrics		
	Backlog	Counting	Pruning
0 - 2,000	< 12000	< 1800	< 1200
2,001 – 10,000	< 10000	< 1700	< 1100
10,001 - 40,000	< 8500	< 1600	< 1000
40,001 - 80,000	< 3500	< 1000	< 900
80,001 - 160,000	< 3000	< 500	< 500

Example Output for SQL Performance Test

The following examples show the test results for some standard DAS configurations:

Entire DB on 8-spindle 2.5” 15K RPM disks in RAID-10

```
Backlog      10965
Counting     1663
Pruning      1071
```

Entire DB on 12-spindle 2.5” 15K RPM disks in RAID-10

```
Backlog      9180
Counting     1652
Pruning      1022
```

Entire DB on 14-spindle 2.5” 15K RPM disks in RAID-10

```
Backlog      8397
Counting     1642
Pruning      850
```

Data + Log + Temp on 14-spindle 2.5” 15K RPM disks in RAID-10
Indexes on Virident FlashMax II 1100 GB card

```
Backlog      3395
Counting     992
Pruning      805
```

Log + Temp on 14-spindle 2.5” 15K RPM disks in RAID-10
Data + Indexes on Virident FlashMax II 1100 GB card

```
Backlog      2670
Counting     474
Pruning      463
```

SQL Server Maintenance

Scheduled Database Tasks

Cb Protection Server will perform its own database maintenance. The following sections describe two maintenance tasks and their schedule:

DailyPruneTask

This task runs every midnight (Cb Protection Server local time). It performs the following actions in sequence:

1. Cleans up obsolete inventory records. This includes records of old files deleted from endpoints as well as inventory files related to deleted computers,
2. Cleans up events and notifications that are below the configured retention threshold,
3. Collects server performance statistics,
4. (Optionally – if configured) Deletes old file catalog files that have 0 prevalence on agent machines,
5. Performs various other data maintenance and cleanup tasks.

DailyPruneTask usually takes between 30 minutes and 3 hours, depending on database size and churn. Console performance might be slower while this task is running.

Database Maintenance

This task runs every Saturday, at 3 AM. It performs the following actions:

1. Defragments database indexes that have a high level of fragmentation (>20%).
2. Creates missing database statistics.
3. Updates obsolete database statistics.

Depending on database size, Database Maintenance can run from 1 to 6 hours. Console performance and other operations might be impacted while this task is running.

Database Growth

Cb Protection server will collect inventories from all computers that are running Cb Protection Agent, and will track any inventory changes. Once a computer is added to the system, it sends its inventory of all “interesting” files to its Cb Protection Server. This can be between 10K and 50K files per endpoint. Once this inventory is sent, only a small number of new files (changes) will be reported on daily bases. This usually includes up to 500 file additions or deletions per endpoint.

Since the Cb Protection Server maintains only the current inventory and only has a limited file deletion/modification history, the database growth is much slower after all endpoints are initialized.

The following table gives more information regarding the growth of specific database files:

Name	File name	Growth	Growth control options
PRIMARY	das.mdf	The growth of this file is not limited since it contains all unique file metadata ever seen in the environment as well as all rules and approvals. Rapid growth of this file after a steady state has been reached could indicate excessive generation of unique files on endpoints.	Optional deletion of files with zero prevalence can be turned on. See the next section, "Pruning File Catalog", for more information.
SECONDARY	das_events.ndf	Growth is limited by event retention thresholds (time and size) set on the System Configuration page	Lowering the thresholds will limit file growth.
ABINST	das_abinst.ndf	Growth is proportional to the number of interesting files on endpoints.	Custom rules can be used to ignore "noisy" files that are not interesting to track.
ABTEMP	das_abtemp.ndf	Growth is proportional to the number of interesting files on endpoints.	Custom rules can be used to ignore "noisy" files that are not interesting to track.
Log file	das_log.ldf	Largest growth occurs during large transactions, most notably during product upgrades.	Changing the server recovery mode or more frequent transaction log backups.
Temp SQL data file	Tempdb.mdf	These files grow mostly during expensive background tasks (like the Daily Prune Task), and should never get larger than 50 GB in total.	SQL Server service restart will reset these files to 0 size.
Temp SQL log file	Tempdb.ldf		

Note that SQL Server only grows its data files and will not reduce them, even as underlying data is deleted. Essentially, a file size represents the maximum watermark of the size that was reached at some point. The only exception to this is the log file, which is controlled by the backup and SQL Server recovery modes.

The following MSDN article describes how to shrink the empty data files. Shrinking data file can have a negative performance impact. Use these steps only when a large amount of data is deleted from the database (e.g., if event retention has been reduced), and you don't expect the database to reuse this space.

http://technet.microsoft.com/en-us/library/ms190757.aspx

Events Growth

The SECONDARY file group (das_events.ndf) will grow based on limits set in the configuration for both public and internal events.

Here is estimated number of daily public and internal events for 1000 agents:

Agent Count	Daily public events	Daily internal events
1	250	100
1,000	250,000	100,000
100,000	25,000,000	10,000,000

A single **public event** requires approx. **0.5 KB** in database, and an **internal event** requires **0.8 KB**. Based on that, here is estimate on final Events filegroup target sizes for different event limits (configurable):

Public events	Internal events	Database file size (das_events.ndf)
1,000,000	100,000	0.6 GB
10,000,000	1,000,000	6 GB
100,000,000	10,000,000	60 GB

Default retention limit for public events is 10M, and for internal events is 100,000.

Note: *Some product features, like Meters and Custom rules, can increase the number of daily events above the expected numbers.*

Pruning the File Catalog

File catalog pruning is disabled by default. You can turn it on by changing the shepherd_config property PurgeAntibodiesPeriodDays. A value of 0 means that files will never be deleted from the catalog.

When this pruning is enabled, Cb Protection will automatically delete all files with a prevalence of 0 (i.e., not existing on any endpoint), after N days from reaching 0 prevalence. Note that if a 0-prevalence file reappears on any endpoint during these N days, its pruning will be canceled.

This feature is useful when a Cb Protection environment has a lot of unique temporary files on the endpoints, causing the PRIMARY filegroup to grow unbound. The downside is that the records of these unique files might still provide some forensic value even after the files themselves are gone.