ControlPoint

Software Version 5.8.0

Database Conversion Guide

Document Release Date: December 2019
Software Release Date: December 2019

Legal notices

Copyright notice

The only warranties for products and services of Micro Focus and its affiliates and licensors (“Micro Focus”) are as may be set forth in the express warranty statements accompanying such products and services. Nothing herein should be construed as constituting an additional warranty. Micro Focus shall not be liable for technical or editorial errors or omissions contained herein. The information contained herein is subject to change without notice.

Documentation updates

The title page of this document contains the following identifying information:

- Software Version number, which indicates the software version.
- Document Release Date, which changes each time the document is updated.
- Software Release Date, which indicates the release date of this version of the software.

To check for updated documentation, visit https://www.microfocus.com/support-and-services/documentation/.

Support

Visit the MySupport portal to access contact information and details about the products, services, and support that Micro Focus offers.

This portal also provides customer self-solve capabilities. It gives you a fast and efficient way to access interactive technical support tools needed to manage your business. As a valued support customer, you can benefit by using the MySupport portal to:

- Search for knowledge documents of interest
- Access product documentation
- View software vulnerability alerts
- Enter into discussions with other software customers
- Download software patches
- Manage software licenses, downloads, and support contracts
- Submit and track service requests
- Contact customer support
- View information about all services that Support offers

Many areas of the portal require you to sign in. If you need an account, you can create one when prompted to sign in. To learn about the different access levels the portal uses, see the Access Levels descriptions.

Contents

Chapter 1: Overview

This document and the supporting SQL script packages are intended to convert the databases in your Micro Focus ControlPoint environment to a more scalable and efficient form. Although strongly recommended for all ControlPoint environments being upgraded, these conversions are not strictly required.

NOTE: A set of 5.8.0 ControlPoint databases where these instructions have not been followed will continue to function. However, none of the benefits detailed here will apply to that environment.

IMPORTANT: This document and the procedures within it are a one-time conversion process for ControlPoint version 5.8.0 environments.

Database naming conventions

By default, the database names in your ControlPoint environment conform to the following naming conventions:

- ControlPoint
- ControlPointAudit
- ControlPointMetaStore
- ControlPointMetaStoreTags
- ControlPointTracking

Benefits

Over the lifetime and growth of a ControlPoint system, particularly at the large enterprise level, overall database resource utilization and performance degrades as the quantity of managed data continues to grow. This database conversion package improves scalability at high volumes by decreasing the rate of degradation.

In addition, it provides the following benefits to all ControlPoint database implementations, regardless of size:

- Reduces the size of the ControlPointMetaStore database, which is one of the largest of the ControlPoint databases. This reduction, in particular to the index component, results in an up to 33% reduction in total database size.

  In addition, the new storage structure of the databases is in smaller, more manageable files. This allows a systems operator to make use of smaller, more independent logical volumes.

- Reduces the storage throughput required for ControlPoint operations because it takes advantage of the concurrent storage channels/volumes usually available to production servers. This results in an up to 66% reduction in required storage throughput for a given workload.
- Separates the structure of the database storage into multiple discrete files. This allows you to more accurately monitor your server for I/O hotspots while under load and to easily relocate component files to additional volumes.

  It allows you to preserve a standard logical internal structure and facilitates future upgrades, even if you performed custom reorganization of the storage files.
- Reduces SQL Server memory utilization. This results in an up to 90% reduction in downward memory pressure under load.
- Adds SQL table and index partitioning. This provides some small (< 10%) query performance improvement in specific query types. However, the major gain in this area is a reduction in necessary SQL index maintenance windows; allowing for more processing hours in a given day.
- Adds maintenance plans to all ControlPoint databases. The maintenance plans can be tailored by the database administrators as needed.

  These scheduled jobs, run by the native SQL Agent, intelligently perform rebuild, re-index, statistic calculation, and index compression tasks automatically and in an optimized fashion for both standard and partitioned objects, utilizing online index maintenance operations when available. For more information on SQL Server Agent jobs, see your SQL Server documentation.

  By default, all of the new scheduled jobs run at 10 pm server time. If desired, you can adjust the nightly schedule times for each database. These start times may be staggered if desired, but it is important to ensure that the jobs are set to run at least once per day.

  For more information, see Advanced database maintenance controls, on page 29.

Requirements

The following details the requirements, in hardware, software and environment resources and in user impact or downtime, necessary to complete the execution of this conversion package.

SQL Server software

A version and edition of SQL Server that supports partitioning and supports a sufficient number of partitions. This currently includes.

> **NOTE:** For each version and edition of SQL Server, you need to apply all currently-available and pushed updates (critical updates and publicly-pushed individual updates) from Windows Update.
>
> For more information about supported versions, see the *ControlPoint Support Matrix*.

SQL Server Management Studio

This process requires access to SQL Server Management Studio either on the server itself (recommended) or remotely from a Windows workstation.

> **IMPORTANT:** Do not use Powershell or SQLCMD to complete the tasks in this Guide.

Permissions

- **SA** or another SQL user account with the **sysadmin** server role assigned, must be used for the conversion.

  In addition, this account (and the account being used to execute the MS SQL Server process) must have the ability to read and write files to the local server disk storage.

  This includes directories chosen for placement of new files and file groups as well as a temporary location to be used for MS SQL Bulk Copy (BCP) workspace.

ControlPoint environment

> **IMPORTANT:** Your ControlPoint environment must be running release 5.4 or later. The conversion package does not support releases earlier than ControlPoint 5.4.
>
> If you are running a version of ControlPoint prior to 5.4:
>
> 1. Download the ControlPoint 5.4 software package from the MySupport portal.
> 2. Upgrade the ControlPoint databases to version 5.4 by following the upgrade procedure documented in the *ControlPoint Installation Guide.*
> 3. Upgrade the ControlPoint databases to version 5.8.0.
> 4. Continue with the conversion tasks in this guide.

Bulk Copy (BCP) workspace

> **IMPORTANT:** The location chosen for the BCP workspace must be of sufficient size to hold the entire data portion (in text form) of any single table. This workspace is emptied during the execution and may be removed after conversion.

The maximum size necessary can be calculated by choosing the widest table, summing the maximum byte count for the column types, and multiplying by the number of rows. The information is simplified in the following example; see TSQL examples, on page 19.

Example: ControlPointMetaStore database, Metadata.Document table

Average size per row	2,752 bytes
Current row count	347,641,154
Necessary temporary BCP workspace to convert this database:	892 GB

Sufficient additional SQL disk storage resources up to 100% of the current database's data portion size will be utilized in the final production configuration.

Micro Focus recommends that the additional space reside on the additional, smaller performant logical volumes targeted for converted database storage.

This information can be retrieved by SQL Server Management Studio (SSMS) by running the `sp_spaceused` command. This space can be returned to the OS (and then removed) after the database conversion is run and data integrity is verified.

Example: ControlPointMetaStore database

database_name	ControlPointMetaStore
database_size	1,007,723.32 MB
unallocated space	14,598.63 MB
reserved	1,016,566,592 KB
data	470,683,552 KB
index_size	545,817,792 KB
unused	65,248 KB

In this example, an additional 450 GB of storage should be available for use during the conversion process.

User impact and downtime

The ControlPoint environment must be completely offline for the duration of this conversion.

> **CAUTION:** All ControlPoint and Micro Focus IDOL services must be stopped before proceeding with the database conversion steps.
>
> Failure to stop all services in your ControlPoint environment may result in failure to successfully convert the database structures using the scripts.

Component processes should be cleanly stopped and disabled in the Windows Services applet. This includes:

- All ControlPoint connectors
- All ControlPoint related Windows processes.
- IIS and W3SVC hosting the ControlPoint user interface

IMPORTANT: When calculating the necessary downtime, include all steps of the conversion process including:

- the initial database backup
- the conversion itself
- the verification of converted data
- the cleanup of backed up data
- the initial execution of the maintenance task.

For large databases, this will be significant. For more information, see Conversion reference, on page 20.

Chapter 2: Run the conversion package

This section details the steps required to execute the conversion package.

For enterprise customers with databases of significant size, run the following tasks on a user acceptance test environment (UAT) that closely matches the production equipment's configuration and capacity.

Most anticipated errors encountered during the process are safely handled or recoverable from, with the investment of extra time. Every attempt is provided to allow administrators to estimate the time and resources required. Micro Focus strongly recommends performing the complete process as a test run with your individual data.

IMPORTANT: This package requires significant time, resources, knowledge and planning to be executed successfully; it must be performed manually. For more information, see Requirements, on page 6.

IMPORTANT: You must perform all tasks to completion.

Before you begin

Perform the following tasks to prepare the environment for running the database conversion packages:

1. On the SQL Server machine, ensure all directories planned for use (for new data files, for BCP, etc.) have the appropriate permissions. This includes:
   a. Set read, write, and change permissions on each directory for the account being used to operate the SQL Server process.

      This is usually either `NT Service\MSSQLSERVER` or `SYSTEM`, but may be a different user account. Check the Services control panel to identify this.

   b. If you are connecting to SQL Server with a non-SQL user account (for example, a Windows account), ensure this account also has read, write, and change access to the new data file directories and the BCP workspace.
2. Allow any executing policy phases to complete.

   NOTE: Ensure all items in the existing policies are in the `executed` or `failed` status, before the upgrade.
3. In the ControlPoint Administration dashboard, disable the Assign Policies and Execute Policies scheduled tasks using the Scheduled Tasks settings. This prevents new policies from being assigned to documents.

   NOTE: Be sure to disable all of the scheduled tasks: Normal, Low and High priority.

4. Ensure that all ingestion jobs are complete.

   NOTE: If ingestion jobs are still running, wait for them to complete before proceeding.

1. Stop all ControlPoint-related processes on all servers that make up the environment.

   CAUTION: All ControlPoint and IDOL services must be stopped before proceeding with the database conversion steps.

   Failure to stop all services in your ControlPoint environment may result in failure to successfully convert the database structures using the scripts.

   - MetaStore service
   - IDOL service
   - OGS service
   - DataAnalysis service
   - Engine service
   - License Server service
   - License Service service
   - Distributed Connector
   - Individual connectors and Connector Framework Services
   - IIS

2. Create full backups of all ControlPoint databases.

   NOTE: Ensure that you have sufficient storage space for the database backups.

   IMPORTANT: This is a critical step and serves as the necessary safety net for recovery in case of failure.

ControlPoint database

This section provides the specific steps to convert the component database, by default named ControlPoint.

To convert the ControlPoint database

1. In SQL Server Management Studio, connect to the SQL Server instance as SA, or equivalent.
2. Select the ControlPoint database.
3. Open the `\ControlPoint\01_Convert_ControlPoint_v1.sql`. Adjust the following:

a. Find and replace all instances of `<ControlPoint_db_name>` with the name of the ControlPointdatabase (usually ControlPoint.

b. Replace the `<bcpPath>` placeholder with a storage location with high I/O performance and adequate space for large table data transfers.

c. Replace the `<username>` placeholder with a SQL user with select and insert permissions to all objects in this database.

d. Replace the `<password>` placeholder with the plain text password of the defined user.

4. Adjust the following parameter if desired:
   - `@largeRowCount` - The maximum number of rows in a table before the BCP backup and copy method is utilized. Default: 20,000,000.
5. Execute the modified script. Progress and results of the operation are returned in the Messages tab.

   TIP: To preserve the record of this execution, save this content to a text file.

6. Open the `\ControlPoint\02_ControlPoint_ConversionCleanup.sql.`
   - Find and replace all instances of the `<server_db_name>` with the name of the ControlPoint database (usually ControlPoint).

     CAUTION: This script is data destructive. If row count verification tests pass, it permanently removes the backup tables created during the conversion process.

     If disk storage space is not immediately required to be returned to the OS, execution of this script may be delayed until full conversion and application function verification is complete.
7. Execute the modified script. Progress and results of the operation are returned in the Messages tab.

ControlPointAudit database

This section provides the specific steps to convert the component database, by default named ControlPointAudit.

To convert ControlPointAudit database

1. In SQL Server Management Studio, connect to the SQL Server instance as SA, or equivalent.
2. Select the ControlPointAudit database.
3. Open the `\ControlPointAudit\01_Convert_ControlPointAudit_v1.sql`. Adjust the following:
   a. Find and replace all instances of `<ControlPointAudit_db_name>` with the name of the ControlPoint database (usually ControlPointAudit).

b. Replace the `<bcpPath>` placeholder with a storage location with high I/O performance and adequate space for large table data transfers.

c. Replace the `<username>` placeholder with a SQL user with select and insert permissions to all objects in this database.

d. Replace the `<password>` placeholder with the plain text password of the defined user.

4. Adjust the following parameter if desired:

   - `@largeRowCount` - The maximum number of rows in a table before the BCP backup and copy method is utilized.

     Default: 20,000,000

5. Execute the modified script. Progress and results of the operation are returned in the Messages tab.

   > **TIP:** To preserve the record of this execution, save this content to a text file.

6. Open the `\ControlPointAudit\02_ControlPointAudit_ConversionCleanup.sql` script.

   a. Find and replace all instances of the `<server_db_name>` with the name of the ControlPoint database (usually ControlPointAudit).

   > **CAUTION:** This script is data destructive. If row count verification tests pass, it permanently removes the backup tables created during the conversion process.
>
> If disk storage space is not immediately required to be returned to the OS, execution of this script may be delayed until full conversion and application function verification is complete.

7. Execute the modified script. Progress and results of the operation are returned in the Messages tab.

ControlPointMetaStore database

This section provides the specific steps to convert the component database, by default named ControlPointMetaStore.

To convert the ControlPointMetaStore database

1. In SQL Server Management Studio, connect to the SQL Server instance as SA, or equivalent.
2. Open the `\ControlPointMetaStore\01_Convert_ControlPointMetaStore_v2.sql` script by adding the new custom columns that you have added.

   Adjust the following:

   a. Find and replace all instances of `<ControlPointMetaStore_db_name>` with the name of the ControlPoint database (usually ControlPointMetaStore).

b. Replace the `<bcpPath>` placeholder with a storage location with high I/O performance and adequate space for large table data transfers.

c. Replace the `<username>` placeholder with a SQL user with select and insert permissions to all objects in this database.

d. Replace the `<password>` placeholder with the plain text password of the defined user.

e. Adjust the following parameter if desired:

- `@largeRowCount` - The maximum number of rows in a table before the BCP backup and copy method is utilized. Default: 20,000,000.

> **IMPORTANT:** If you added additional custom properties to your MetaStore, you will have to modify the `01_Convert_ControlPointMetaStore_v2.sql script` to keep your custom properties.
>
> For more information, see Modify the conversion script for environments with additional custom properties, below.

3. Execute the modified script. Progress and results of the operation are returned in the **Messages** tab.

> **IMPORTANT:** This script generates a table named `ControlPointMetadata.Additional_backup`. If you should need to re-run this script for any reason, such as if you had forgotten to update it with custom properties, manually delete this table first. Otherwise, the script fails because the table exists.

> **TIP:** To preserve the record of this execution, save this content to a text file.

4. Open the `\ControlPointMetaStore\02_ControlPointMetaStore_ConversionCleanup.sql`.

- Find and replace all instances of the `<server_db_name>` with the name of the ControlPoint database (usually ControlPointMetaStore).

> **CAUTION:** This script is data destructive. If row count verification tests pass, it permanently removes the backup tables created during the conversion process.
>
> If disk storage space is not immediately required to be returned to the OS, execution of this script may be delayed until full conversion and application function verification is complete.

5. Execute the modified script. Progress and results of the operation are returned in the Messages tab.

Modify the conversion script for environments with additional custom properties

If you added additional custom properties to your MetaStore, you will have to modify the `01_Convert_ControlPointMetaStore_v2.sql` script to keep your custom properties.

To modify the conversion script

1. Perform steps 2 a through e from the procedure above to edit the `01_Convert_ControlPointMetaStore_v2.sql` script.
2. Edit the **[ControlPointMetadata].[Additional]** table creation section of the `01_Convert_ControlPointMetaStore_v2.sql` script by adding the new columns that you have added.

   NOTE: Micro Focus recommends that you get the definition for the new columns from SQL Server.

   Example

```
--Object is partitioned, need to check for scheme existance
IF EXISTS(SELECT * FROM sys.partition_schemes WHERE name = 'ps_binary_eight_fg_
ControlPointMetadata_data' )
BEGIN
     IF NOT EXISTS (SELECT * FROM sys.objects WHERE object_id = OBJECT_ID(N'
[ControlPointMetadata].[Additional]') AND type in (N'U'))
     BEGIN
             CREATE TABLE [ControlPointMetadata].[Additional](
                         [RepositoryId] [int] NOT NULL,
                         [DocKey] [binary](8) NOT NULL,
                         [HPRMDataSet] [nchar](2) NULL,
                         [ComparisonField] [nvarchar](64) NULL,
                         [MatchWithinArchive] [binary](8) NULL,
                         [MatchArchive] [binary](8) NULL,
                         [FileType] [nvarchar](32) NULL,
                         [HPRMClassification] [nvarchar](max) NULL,
                         [HPRMContainer] [nvarchar](max) NULL,
                         [SPUUID] [nvarchar](128) NULL,
                         [SPSiteURL] [nvarchar](max) NULL,
                         [SPListURL] [nvarchar](max) NULL,
                         [TrimURLLocationHash] [binary](8) NULL,
                         [ImportErrorCode] [int] NULL,
                         [DocumentDateCreated] [datetime] NULL,
                         [CustomColumnName] [datatype] DEFAULT if any,
            CONSTRAINT [ControlPointMetadata_Additional_PK] PRIMARY KEY
NONCLUSTERED
             (
                          [DocKey] ASC,
                          [RepositoryId] ASC
             )WITH (PAD_INDEX = OFF) ON ps_binary_eight_fg_
ControlPointMetadata_index(DocKey)
             ) ON ps_binary_eight_fg_ControlPointMetadata_data(DocKey)
             Print '    Table [ControlPointMetadata].[Additional] created.'
   END
END
```

```
--Regular version if PS doesn't exist.
IF NOT EXISTS (SELECT * FROM sys.objects WHERE object_id = OBJECT_ID(N'
[ControlPointMetadata].[Additional]') AND type in (N'U'))
BEGIN
          CREATE TABLE [ControlPointMetadata].[Additional](
                         [RepositoryId] [int] NOT NULL,
                         [DocKey] [binary](8) NOT NULL,
                         [HPRMDataSet] [nchar](2) NULL,
                         [ComparisonField] [nvarchar](64) NULL,
                         [MatchWithinArchive] [binary](8) NULL,
                         [MatchArchive] [binary](8) NULL,
                         [FileType] [nvarchar](32) NULL,
                         [HPRMClassification] [nvarchar](max) NULL,
                         [HPRMContainer] [nvarchar](max) NULL,
                         [SPUUID] [nvarchar](128) NULL,
                         [SPSiteURL] [nvarchar](max) NULL,
                         [SPListURL] [nvarchar](max) NULL,
                         [TrimURLLocationHash] [binary](8) NULL,
                         [ImportErrorCode] [int] NULL,
                         [DocumentDateCreated] [datetime] NULL,
                         [CustomColumnName] [datatype] DEFAULT if any,
                  CONSTRAINT [ControlPointMetadata_Additional_PK] PRIMARY KEY
NONCLUSTERED
                  (
                           [DocKey] ASC,
                           [RepositoryId] ASC
                  )WITH (PAD_INDEX = OFF) ON [fg_ControlPointMetadata_index]
                  ) ON [fg_ControlPointMetadata_data]
                  Print '    Table [ControlPointMetadata].[Additional] created.'
END
```

3. If you added any indexes for the new column in the **[ControlPointMetadata].[Additional]** table, you must add them to the index creation section.

   > **NOTE:** Replace the **bolded** portion with the name of the index and the appropriate column name.

   The index section begins with the following line:

   ```
   -- create indexes functions and views
   ```

   Example

   ```
   -- create indexes functions and views

   ...

   IF NOT EXISTS (SELECT name FROM sys.indexes WHERE object_id = OBJECT_ID(N'
   [ControlPointMetadata].[Additional]')
   ```

```
AND name = 'ControlPointMetadata_Additional_IDX_IndexName')
BEGIN
       CREATE NONCLUSTERED INDEX [ControlPointMetadata_Additional_IDX_
IndexName'] ON
       [ControlPointMetadata].[Additional]
        (
                        [CustomColumnName] ASC
        ) WITH (PAD_INDEX = OFF) ON [fg_ControlPointMetadata_index]
        Print '    Non Clustered Index ControlPointMetadata_Additional_IDX_
IndexName ON
       [ControlPointMetadata].[Additional] created.'
END
```

4. Continue with steps 3 through 5 from the conversion procedure above.

ControlPointTracking database

This section provides the specific steps to convert the component database, by default named ControlPointTracking.

To convert the ControlPointTracking database

1. In SQL Server Management Studio, connect to the SQL Server instance as SA, or equivalent.
2. Open the `\ControlPointTracking\01_Convert_ControlPointTracking_v1.sql` script.

   Adjust the following:

   a. Find and replace all instances of `<ControlPointTracking_db_name>` with the name of the ControlPoint database (usually ControlPointTracking).

   b. Replace the `<bcpPath>` placeholder with a storage location with high I/O performance and adequate space for large table data transfers.

   c. Replace the `<username>` placeholder with a SQL user with select and insert permissions to all objects in this database.

   d. Replace the `<password>` placeholder with the plain text password of the defined user.
3. Adjust the following parameter if desired:

   - `@largeRowCount` - The maximum number of rows in a table before the BCP backup and copy method is utilized. Default: 20,000,000.
4. Execute the modified script. Progress and results of the operation are returned in the Messages tab.

   > **TIP:** To preserve the record of this execution, save this content to a text file.
5. Open the `\ControlPointTracking\02_ControlPointTracking_ConversionCleanup.sql`

script.

- Find and replace all instances of the `<server_db_name>` with the name of the ControlPoint database (usually ControlPointTracking).

> **CAUTION:** This script is data destructive. If row count verification tests pass, it permanently removes the backup tables created during the conversion process.
>
> If disk storage space is not immediately required to be returned to the OS, execution of this script may be delayed until full conversion and application function verification is complete.

6. Execute the modified script. Progress and results of the operation are returned in the Messages tab.

Appendix A: TSQL examples

The following section includes the TSQL examples that you can use to gather information from your ControlPoint environment.

Calculate expected data size of a BCP output

```
select sum(a.max_length) as AvgRowSizeInBytes from
( select c.name,
       case
               when c.max_length < 0 then 1024
               --Assume unlimited length fields to be an average length,
               1KB in this case when ty.name like '%var%' then (c.max_length/4)
               --When a variable width field is found, take a fraction
               of the maximum length (25% in this example)
               --or to be more accurate calculate the current average
               length of data currently in the column
               else c.max_length
       end as max_length
       from sys.columns c (nolock)
       inner join sys.tables t (nolock) on c.object_id = t.object_id
       inner join sys.types ty (nolock) on c.user_type_id = ty.user_type_id
       where t.name = 'Document' --Table Name
) as a
```

View index maintenance records

```
--20 Most Fragmented Indexes/Partitions
select top 20 t.table_schema, t.table_name, tim.IndexName, tim.PartitionNumber,
       tim.AvgFragmentationInPercent, tim.lastActionDate, timc.ActionDescription
       from dbo.table_indexMaintenance tim (nolock)
       inner join dbo.tables t (nolock) on tim.table_id = t.table_id
       inner join dbo.table_indexes ti (nolock) on tim.table_id = ti.table_id
       inner join dbo.table_indexMaintenance_code timc (nolock) on timc.LastActionType
=
       tim.LastActionType
       where t.table_name not in ('table_indexes', 'table_indexMaintenance',
       'table_indexMaintenance_code', 'tables', 'unique_ids')
       order by tim.avgFragmentationInPercent desc, tim.lastActionDate desc
```

Appendix B: Conversion reference

This appendix provides reference information to support improvement claims and to provide data to facilitate conversion execution planning.

Conversion

Size reference information focuses exclusively on the ControlPointMetaStore database. The other databases' storage size, as well as the time and resources to convert them, are insubstantial in comparison.

Total number of stored documents: 347,641,154

Reference SQL Server

CPU	2x Intel Xeon E5 2660 @ 2.20GHz, Hyperthreading Enabled (Total: 16 core/32 thread)
RAM	128G DDR3 (SQL permitted to use 124G)
System Disk	1x 450G HDD
Tempdb Disk	7x 900G 2x HDD RAID0
Database Disk	3x 1500G SSD

Disk I/O reference (concurrently measured by CrystalDiskMark)

Report ID	Sequential Read	Sequential Write	Random Read 512K	Random Write 512K	Random Read 4K (QD=1)	Random Read 4K (QD=1) IOPS	Random Write 4K (QD=1)	Random Write 4K (QD=1) IOPS	Random Read 4K (QD=32)	Random Read 4K (QD=32) IOPS	Random Write 4K (QD=32)	Random Write 4K (QD=32) IOPS
system	189.633	151.879	79.424	142.545	0.672	164.1	5.313	1297.2	2.050	500.5	5.257	1283.4
tempdb	311.058	290.263	70.920	142.474	0.820	200.3	6.430	1569.9	4.116	1005	7.004	1710
database	384.987	448.749	329.368	494.759	14.115	3446	67.043	16367.9	85.023	20757.7	82.046	20030.8

5.8.0 database size pre-conversion

database_name	**database_size**	**unallocated space**	
ControlPointMetaStore	1007723.32 MB	14598.63 MB	
Reserved	**data**	**index_size**	**unused**
1016566592 KB	470683552 KB	545817792 KB	65248 KB

5.8.0 database size post-conversion

database_name	database_size	unallocated space	
ControlPointMetaStore	631867.57 MB	1423.22 MB	
Reserved	**data**	**index_size**	**unused**
645181920 KB	471001296 KB	172662280 KB	1518344 KB

Execution times

00:59:53	Drop all dependent objects, Rename to create backup tables, Create replacement tables, Create temporary indexes on backup tables
31:13:00	Restore all data to new tables (Filegroups and partitions, even distribution)
03:51:00	Recreate non-clustered indexes (Filegroups and partitions, even distribution)
00:47:00	Recreate all remaining constraints
00:01:00	Recreate all functions and views
36:50:53	Total

Performance

Metrics in this section normalized to the same data set, the same SQL Server hardware, etc. and are intended to be directly comparable and of substantial enough size to indicate likely real world performance.

The ControlPoint configuration driving this collection is also static and separate from the SQL Server.

Concurrent collections: 4

Collection size (documents): 1,000,000

Total documents per test iteration: 4,000,000

Reference SQL Server

CPU	2x Intel Xeon E5620 @ 2.40GHz, Hyperthreading Enabled (Total: 8 cores/16 threads)
RAM	32G DDR3 (SQL Server is permitted to use 16G)

Reference SQL Server, continued

System Disk	1x 300G 2x HDD RAID1
Tempdb & Database disks	3x 300G 2x HDD RAID1 1x 900G 6x HDD RAID0+1

Disk I/O reference (concurrently measured by CrystalDiskMark)

Report ID	Sequential Read	Sequential Write	Sequential Read (QD=32)	Sequential Write (QD=32)	Random Read 4K (QD=1)	Random Read 4K (QD=1) IOPS	Random Write 4K (QD=1)	Random Write 4K (QD=1) IOPS	Random Read 4K (QD=32)	Random Read 4K (QD=32) IOPS	Random Write 4K (QD=32)	Random Write 4K (QD=32) IOPS
System R1	165.477	195.864	245.840	189.610	1.232	300.8	12.695	3099.4	4.895	1195.1	11.495	2806.4
Database R0+1	467.257	440.631	510.922	370.264	1.748	426.8	19.772	4827.1	17.086	4171.4	20.711	5056.4
Database R1	272.000	241.378	250.913	215.737	1.924	469.7	19.153	4676.0	10.662	2603.0	16.249	3967.0

Detailed metric information

This section presents detailed information captured during the collection operations. Quick guidances for each report and what it indicates are as follows:

- **Disk Performance** – The percentage of availability of the logical volume reported to Windows. Higher is better. The average value over the measurement window calculates both this value and the average read and write queue depth on the volume. Lower is better.

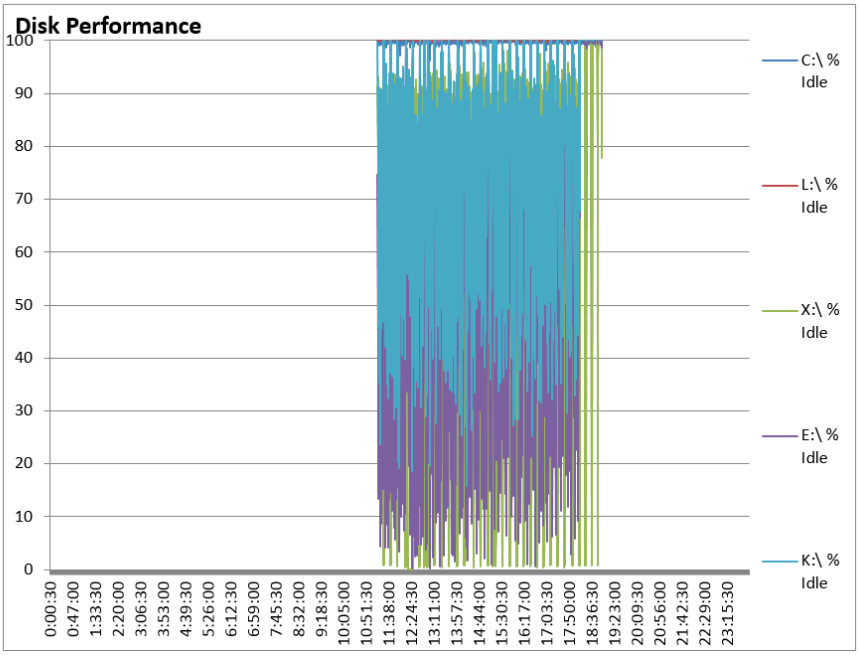
  A fully utilized disk will show queue depth approaching 1 when considered over time. A value above 1 indicates that improving I/O throughput of the volume would improve performance. The delta above 1 can be used as a relative measure of over-utilization of the logical volume.

- Locks (ms) – The time for SQL to acquire and hold necessary locks, of all kinds, to perform the requested operations. Lower is better.

- Average Wait Time (ms) – The amount of time required to acquire resources to perform requested operations. Lower is better. Displayed are times to acquire the necessary locks, the time to acquire the network socket for the data, and the time to acquire the necessary pages (disk).

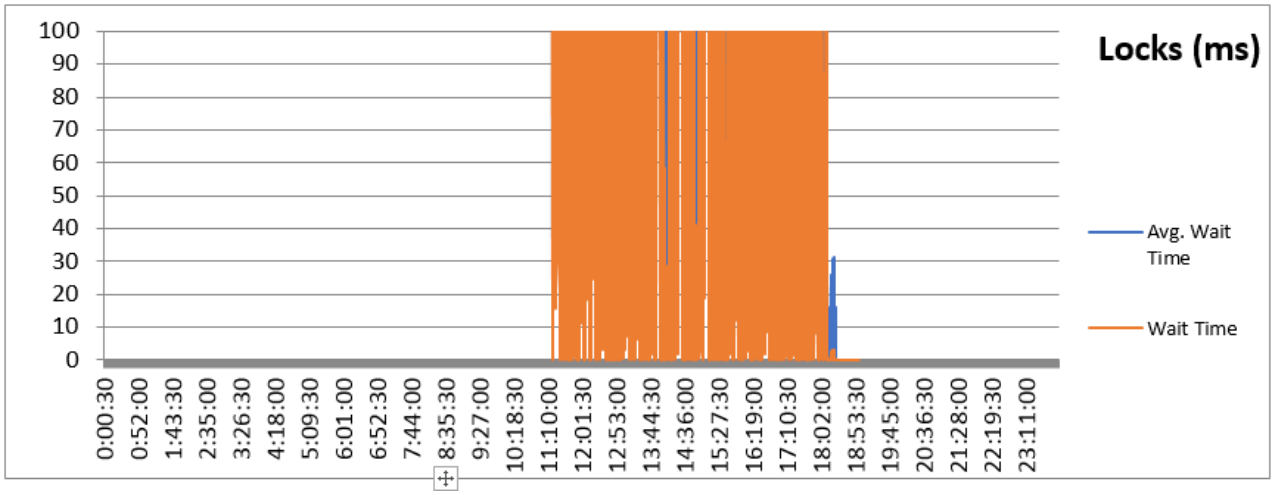
- SQL Page Life Expectancy (s) – The amount of time any given page can be expected to reside in memory. Higher is better. This measure is used to consider memory pressure in SQL. Anything below 600 indicates that memory available to SQL is significantly reducing performance.

5.8.0 database metrics pre-conversion



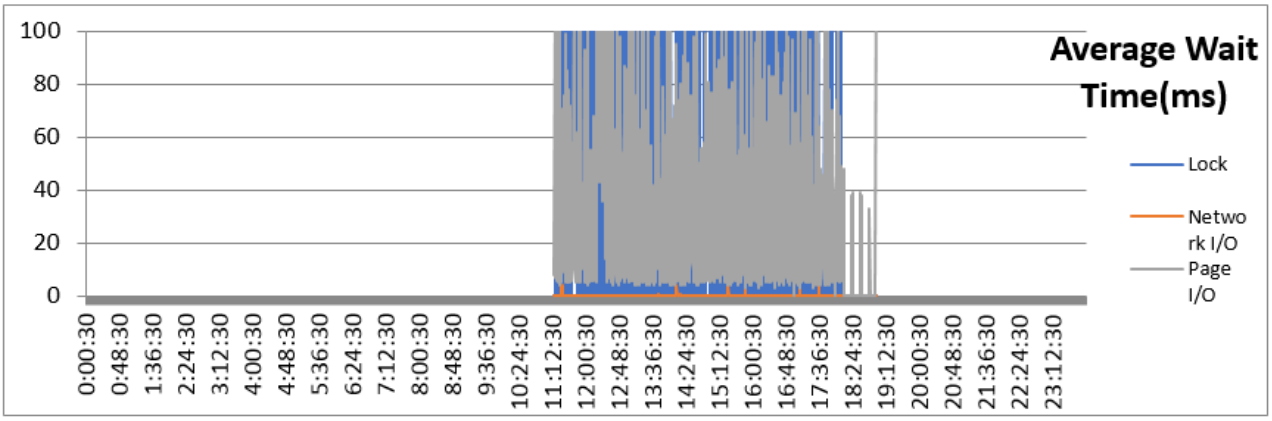
Averages

99.53 C:\ % Idle	0.01 C:\ QD
100.10 L:\ % Idle	0.00 L:\ QD
69.02 X:\ % Idle	25.26 X:\ QD
56.15 E:\ % Idle	18.18 E:\ QD
75.37 K:\ % Idle	11.88 K:\ QD



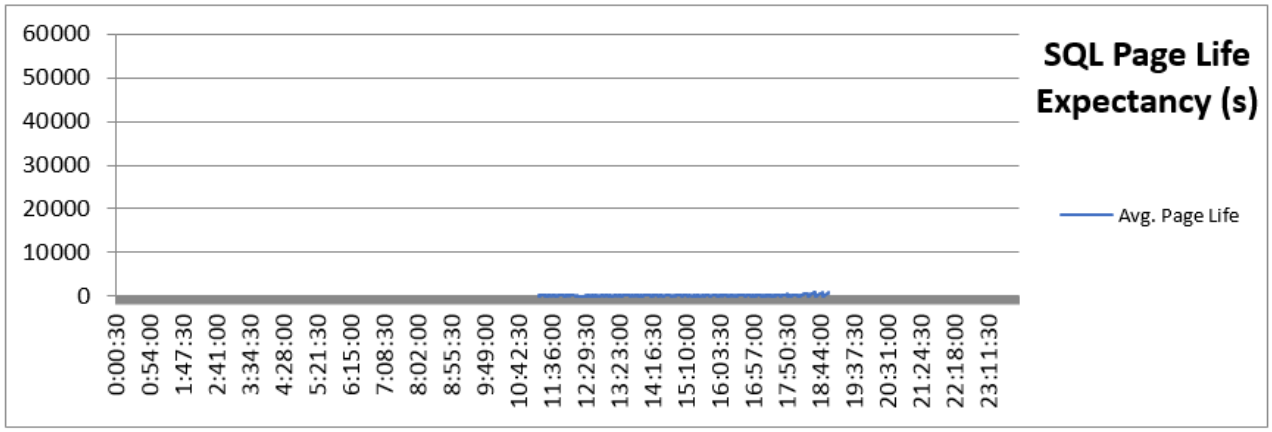
Averages

4433.80	Average wait time
2326.94	Wait time



Averages

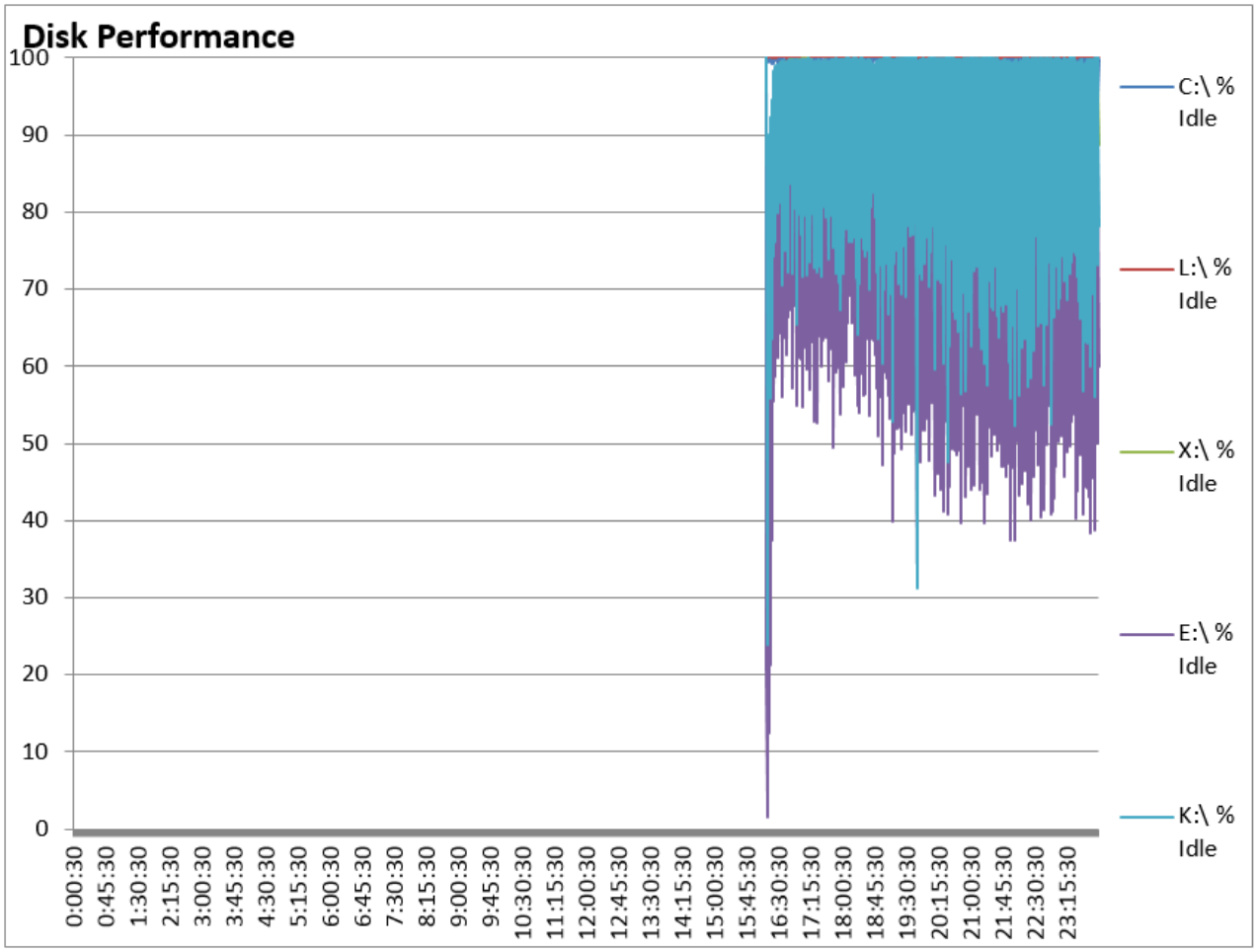
1117.45	Lock
0.03	Network I/O
33.51	Page I/O



Averages

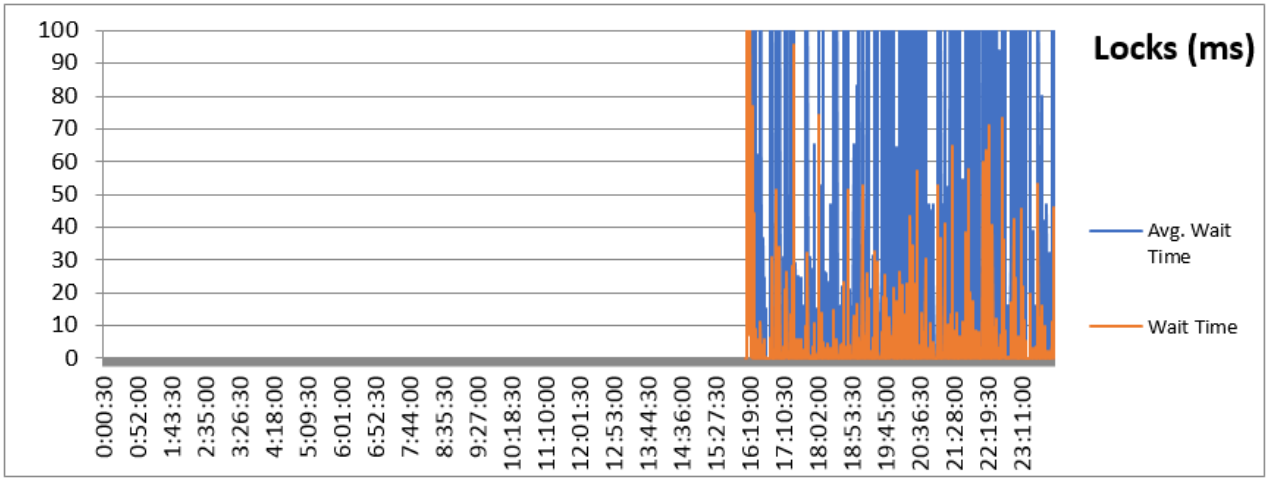
174.54	Average page life

5.8.0 database metrics post-conversion



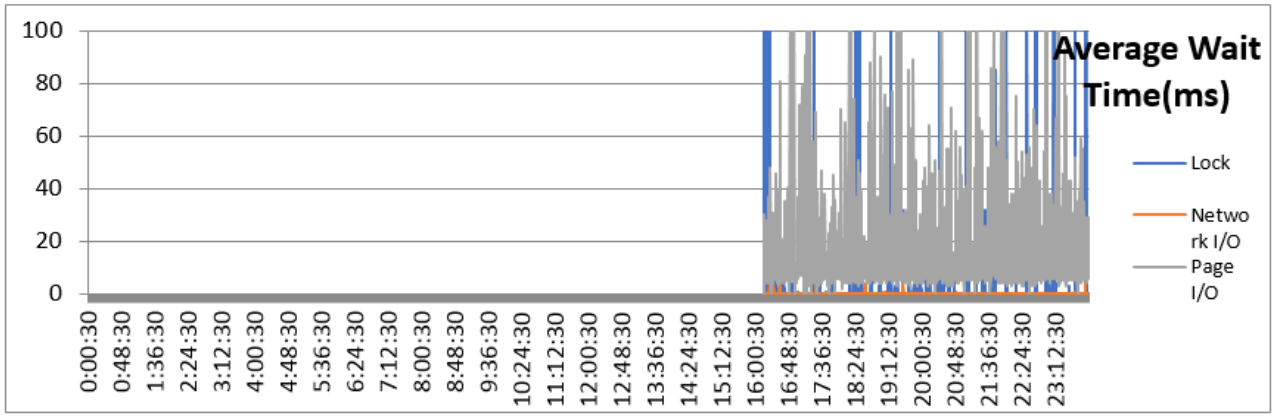
Averages

99.63 C:\ % Idle	0.00 C:\ QD
100.12 L:\ % Idle	0.00 L:\ QD
94.98 X:\ % Idle	0.21 X:\ QD
78.36 E:\ % Idle	19.50 E:\ QD
88.56 K:\ % Idle	10.16 K:\ QD



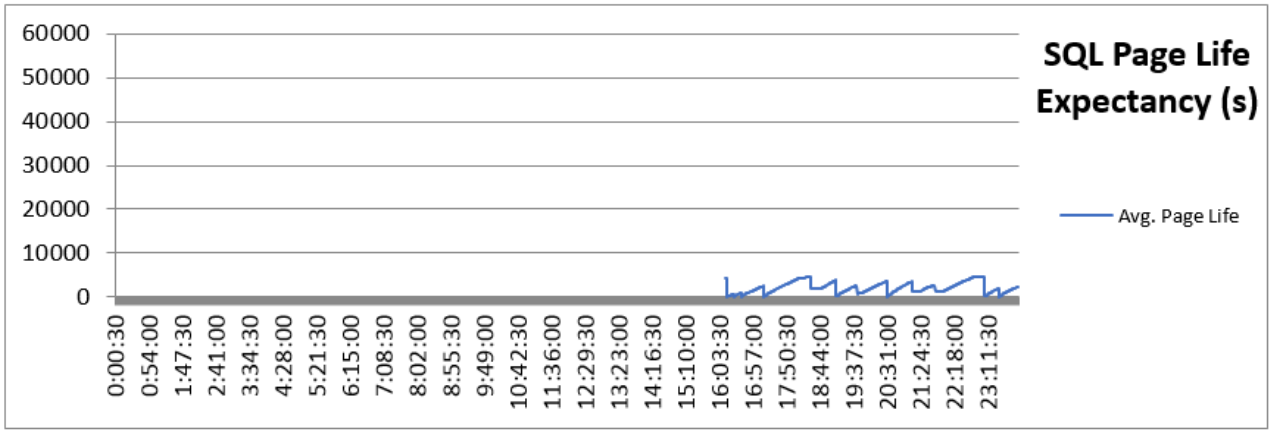
Averages

51.87	Average wait time
4.91	Wait time



Averages

15.85	Lock
0.03	Network I/O
17.08	Page I/O



Averages

2064.48	Average page life

Appendix C: Advanced database maintenance controls

This section details the maintenance jobs' advanced configuration options available in each ControlPoint database after the conversion process has been completed.

Each database's maintenance job utilizes the same structure and offers the same level of control. Each job is configured and scheduled separately and records information in the associated database. The SQL Agent job history for each job is also maintained separately.

Overview

Database maintenance is critical to the ongoing performance of any SQL database. When databases are small or lightly utilized, this is often overlooked or can be performed in a simple manner by a DBA.

However, when databases are heavily utilized and grow to enterprise scale, effectively maintaining them becomes essential. In addition, as the maintenance tasks may require significant time to execute, it becomes necessary to be able to selectively target objects requiring operations so that maintenance can be completed in the available windows to minimize impact to application operation.

This conversion package adds the use of table and index partitioning to the databases, when the SQL server hosting them support it. This applies to Enterprise Editions and SQL 2016 SP1 Standard Edition. Although critical to improving index maintenance task performance, they require a more complex process to efficiently perform this operation.

By default, the conversion process adds standard mechanisms to all ControlPoint databases to accomplish this and requires no further configuration. However, a DBA or system administrator may further tune them to achieve even greater levels of control, tailored to their environment, using the frameworks that are deployed with this conversion.

Basic operations

After completion of the conversion package for a given database, two new SQL agent jobs appear in the **SQL Server Agent->Jobs** area of SQL Server Management Studio.

- `<databaseName>_db_maint_3.0`

  This job is scheduled to run every night at 10 PM SQL Server time. On its execution, it gathers all index and partition fragmentation levels on all current objects in the database and records them to its working tables. This includes adding new objects found to those structures, removing objects that are no longer present, and so on. It then analyses each index and partition, to determine if it is intended to be operated on that day of the week.

By default, all objects, when first detected, are scheduled for operation Tuesday, Thursday, and Saturday.

If an object is sufficiently fragmented, the most efficient maintenance action will be performed against it. Below 2% fragmentation, no action is taken. Between 2 and 30%, a reorganize action is taken. Above 30%, a rebuild operation is taken.

After completing all index operations, all statistics (both index bound and otherwise) are recalculated on all objects.

- `<databaseName>_db_maintain_all`

  This job is not scheduled and is only to be run manually by a DBA or system administrator. When executed, it analyzes all indexes and partitions regardless of their configured target days. The remainder of the logic is unchanged from the regularly scheduled job.

Advanced operations

You can make further adjustments to the scheduled of the maintenance jobs to further tailor the behavior to what is appropriate to your ControlPoint environment.

Change the job schedule

You can modify the schedules with SQL Server Management Studio to execute at different times of the day. This is often used to stagger the maintenance operations of the different databases, permitting SQL server to use its resources in a more sequential fashion. As maintenance jobs are generally very heavy on memory and disk I/O; this can often be advantageous.

> **IMPORTANT:** Do not adjust the schedule so much that a job is prevented from operating at least once every day. To control which objects receive consideration on any given day of the week, see Change the objects targeted each day.

For more information on modifying scheduled jobs in SQL Server Management Studio, see your SQL Server documentation.

Change the objects targeted each day

You can schedule any index or partition object with more or less frequency to best fit into nightly maintenance windows. This can be especially useful as some indexes contain vastly different amounts of data (and so require very different times to complete maintenance tasks) or fragment at different rates based on incoming volume.

By default, all indexes and partitions have operations performed against them on Tuesday, Thursday, and Saturday.

This information is held in the **`dbo.table_indexes`** table.

To adjust the **`dbo.table_indexes`** table

1. Update the **maintenance_day** column.

   Select multiple days by appending the abbreviation codes together. For example, Su = Sunday, Mo = Monday, Tu = Tuesday, We = Wednesday, Th = Thursday, Fr = Friday, Sa = Saturday.

   Example

   Change the schedule of the `CPGlobalSettings_PK` index to be done every day:

```
update dbo.table_indexes
       set maintenance_day = 'SuMoTuWeThFrSa'
       where index_name = 'CPGlobalSettings_PK'
```

Change the logic used to identify necessary index actions

You can control the identification criteria used to determine if an index requires operation, by modifying the `[dbo].[db_index_maint]` stored procedure.

`@reindex_max_frag` – When fragmentation is above this percentage, the index will be rebuilt. Default: 30

`@reorg_max_frag` – When fragmentation is between this percentage and `@reindex_max_frag`, the index will be reorganized. Default: 2

`@PCNT` – The percentage of rows in a table that will be sampled when updating statistics, unless this percentage of the estimated rowcount would result in a larger number of rows than `@maxRowsForStat`. Default: 35

`@maxRowsForStat` – The maximum number of rows in a table sampled to update statistics. Default: 1,000,000

Compress or decompress an index

One of the premium features in SQL Server Enterprise edition is index compression. Although not enabled for any ControlPoint object by default, a DBA may wish to make use of it.

To compress or decompress an index using the maintenance job

- Change the `data_compression` value in the **`dbo.table_indexes`** table

  where

  0 = No Compression

  1 = Row Compression,

  2 = Page Compression.

  The next day that index is targeted, the index will be rebuilt with or without compression.

  Example

```
update dbo.table_indexes
       set data_compression = 1
       where index_name = 'CPGlobalSettings_PK' and data_compression = 0
```

View recorded fragmentation levels and last actions

You may be called upon to examine the fragmentation levels of indexes so to understand and address queries or inserts that may be performing sub-optimally. This information can then be used to adjust day of week targets for specific indexes to have them maintained more frequently as required by the implementation.

> **NOTE:** Collection of this information in real-time may not be feasible, due to impact to concurrent database operations being performed when the application is active. The maintenance jobs record and utilize this information in the following table objects.
>
> For more information on how to read them, see TSQL examples, on page 19.

Send documentation feedback

If you have comments about this document, you can contact the documentation team by email. If an email client is configured on this system, click the link above and an email window opens with the following information in the subject line:

Feedback on Database Conversion Guide (Micro Focus ControlPoint 5.8.0)

Add your feedback to the email and click **Send**.

If no email client is available, copy the information above to a new message in a web mail client, and send your feedback to swpdl.controlpoint.docfeedback@microfocus.com.

We appreciate your feedback!