Windows* Getting Started Guide

Intel® QuickAssist Technology

Hardware Version 2.0

Production Release

October 2023

Document Number: 337758

CONTENTS

1 Introduction **2**
- 1.1 About This Document . . . 2
- 1.2 Features Implemented . . . 2
- 1.3 Package Release Structure . . . 3
- 1.4 System Configuration . . . 3
  - 1.4.1 Configuring BIOS . . . 3
  - 1.4.2 Configuring Windows* . . . 4

2 Software Installation **5**
- 2.1 Unpacking the Driver Package . . . 5
- 2.2 Installation Overview . . . 5
- 2.3 Uninstall Overview . . . 6
- 2.4 Configure Acceleration Software . . . 6
- 2.5 Uninstall Driver Package . . . 6

3 Compression Test Application **7**
- 3.1 Parcomp Command-Line Options . . . 7
- 3.2 Parcomp Examples . . . 9
  - 3.2.1 Compress and Decompress via 'qat' . . . 9
  - 3.2.2 Compress/Decompress Using Gzip Extended Header . . . 10
  - 3.2.3 Compress/Decompress Using Xpress* . . . 11
  - 3.2.4 Compress/Decompress Performance . . . 11

4 Cryptography Test Application **13**
- 4.1 Cngtest Examples . . . 13
  - 4.1.1 RSA 3072-bit Key . . . 13
  - 4.1.2 ECDH using Curve25519 . . . 14
  - 4.1.3 ECDSA using NistP521 . . . 14
  - 4.1.4 RSA 2048-bit Key Performance . . . 15

Intel® QuickAssist Getting Started Guide for Windows*

CHAPTER

ONE

INTRODUCTION

1.1 About This Document

This getting started guide documents the instructions to obtain, install, and exercise the Intel® QuickAssist Technology (Intel® QAT) Windows* software for the Hardware Version 2.0 package.

In this document, for convenience:

- Software package is used as a generic term for the Intel® QAT Software Package.
- Acceleration driver is used as a generic term for the software that allows the Intel® QAT Software Library APIs to access the Intel® QAT Endpoint(s).

Note: Please refer to the Windows* QAT Release Notes for a list of supported devices and/or platforms.

1.2 Features Implemented

Implemented features are listed in Windows* QAT Release Notes or Windows* QAT Technical Guide.

1.3 Package Release Structure

After unpacking the zip file, the directory should contain the following:

Table 1.1:: Windows* Package Release Structure

Files/Directory	Comments
QATx.x.<version>.zip	Top-level Intel® QAT package
./filelist	List of files in this package
./versionfile	The version of this package; this should match the package version
./QuickAssist	Primary folder containing the files necessary for installation, development, license files, and the README.txt
./QuickAssist/Compression	The Windows* QATzip files necessary for data compression development and the compression sample application (parcomp.exe)
./QuickAssist/Crypto	The binaries necessary to register the QAT device for CNG and the crypto sample application (cngtest.exe)
./QuickAssist/Driver	The Windows* QAT base driver and counter manifest file
./QuickAssist/Firmware	The Windows* QAT firmware files and license manifest file
./QuickAssist/Setup	The Windows* QAT driver installation binary file (QatSetup.exe)

1.4 System Configuration

This section describes the process of configuring the system prior to the Intel® QuickAssist Technology (Intel® QAT) driver installation.

Important: Make sure that the platform and/or CPU SKU has the QAT hardware.

1.4.1 Configuring BIOS

For Intel® Xeon® 4th- and 5th-Generation Scalable Processor with Intel® QAT Gen4 and Gen4m, depending on the specific SKU, there can be up to 4 QAT endpoints per socket. It may be possible to disable individual QAT endpoints at the BIOS level by following the instructions below:

Note: The BIOS configuration may differ depending on the platform used. This step is not necessary in a virtual environment.

1. Enter BIOS setup.
2. Navigate to the following path where <n> corresponds to the socket containing the QAT endpoint(s) to be disabled:

```
EDKII Menu > Socket Configuration > IIO Configuration > IOAT Configuration >
Sck<n> > IOAT Configuration
```

1. Update the CPM value to Disable for each QAT endpoint to be disabled for each socket.
2. Save changes.
3. Reboot the system

1.4.2 Configuring Windows*

For the supported Windows* Operating Systems, there should be no additional steps or packages required to use the QAT devices.

CHAPTER

TWO

SOFTWARE INSTALLATION

2.1 Unpacking the Driver Package

The driver package is in a zip package. By default, all supported Windows* Operating Systems can extract this driver package.

Note: To install the QAT drivers, you must have Administrator privileges.

2.2 Installation Overview

The installer requires the Microsoft* VC Redistributable. If the redistributable is not available, it will automatically be installed. The recommended installation method for Windows* is to use the QatSetup.exe binary. The QatSetup.exe binary can be found in the relative directory mentioned in *Package Release Structure*.

Note: For other methods of installation of the QAT driver in Windows*, see the Windows* QAT Technical Guide.

For virtualization (SR-IOV) support without QAT Host services, select the option to install as a "virtualization host" in the installation program. A system restart may be required at the end of the installation in order to fully enable virtualization support.

Important: When doing a Plug and Play (PnP) install of the Windows* QAT driver, only the QAT device driver will be functional. To use hardware accelerated Compression and Crypto services, those services need to be manually started.

To verify that the driver is installed properly, check the Device Manager for devices under 'Security Accelerator'. The following table shows the corresponding QAT device with its respective Device ID.

Table 2.1:: Windows* QAT HW 1.7/1.8 Device ID

QAT Accelerator	Device ID
Intel® 4xxx Accelerator	4940
Intel® 4xxx Accelerator Virtual Function	4941
Intel® 401xx Accelerator	4942
Intel® 401xx Accelerator Virtual Function	4943

2.3 Uninstall Overview

To uninstall the Intel® QuickAssist Accelerator software:

- Open "Programs and Features" from the Control Panel application
- Click on the installed application "Intel® QuickAssist Technology 1.11.0.xxxx"
- Choose Uninstall
- Reboot

Important: When doing a driver uninstall using the Windows Device Manager GUI sequentially, the QAT device driver may not be properly removed when multiple QAT devices are present.

2.4 Configure Acceleration Software

For detailed overview on how to configure the QAT device in Windows*, see the Windows* QAT Technical Guide.

2.5 Uninstall Driver Package

From the Windows* Desktop, the QAT driver can be uninstalled from the Control Panel via the "Uninstall a Program" submenu. Select the entry with Name "*Intel® QuickAssist Technology*".

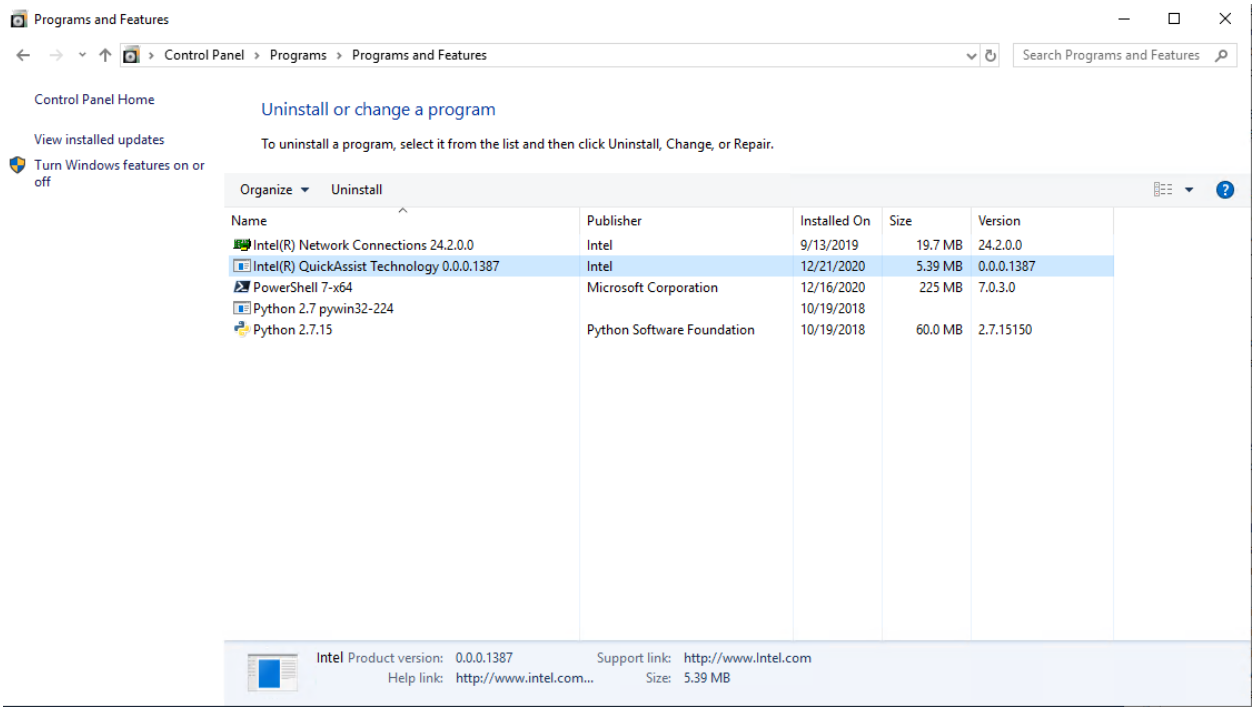

Figure 2.1:: Windows* QAT Uninstallation

CHAPTER

THREE

COMPRESSION TEST APPLICATION

This package comes with a tool called 'parcomp' to test the performance of the Intel® QuickAssist Technology accelerator. Parcomp has been built using the QATzip API and library found within the same driver package. The parcomp binary can be found in the following folder upon completion of the installation:

Program FilesIntelIntel(R) QuickAssist TechnologyCompression

Parcomp can be used to measure and report the rate at which compression and decompression operations are performed using the accelerator as well as those operations performed by the default services offered by the system OS.

Note: It is recommended to use parcomp as Administrator.

The throughput reported by parcomp is measured around the QATzip API call.

3.1 Parcomp Command-Line Options

To see the list of supported command-line options, run either of the following commands:

```
parcomp.exe

parcomp.exe -h
```

The output will be as follows:

```
Usage: parcomp.exe -i <srcFilename> -o <dstFilename> [options]

Required options:

   -i srcFilename        Input (source) filename
   -o dstFilename        Output (destination) filename

Optional options can be:

   -b [cold|warm]        Use cold buffer or not.
   -p providerName       Specifies the provider (implementation).
                        Options include:
                           'qat' - QuickAssist accelerated DEFLATE* algorithm
                                 with 4 byte headers.
                           'qatzlib' - QuickAssist accelerated DEFLATE*
                                       algorithm with zlib* header.
```

(continues on next page)

(continued from previous page)

```
                          'qatms' - Deprecated QuickAssist accelerated
                                  DEFLATE* algorithm with MSZIP* header
                                  header. Only decompression supported.
                          'qatgzip' - QuickAssist accelerated DEFLATE*
                                     algorithm with Gzip* header.
                                     Note: With qatgzip provider, options
                                     -k -t -Q cannot be used in
                                     decompression direction.
                                     The operation will be executed in a
                                     single thread with SW implementation.
                                     Parcomp application cannot process
                                     Gzip* source buffers bigger than 999MB
                                     and the number of iterations is 1.
                          'qatgzipext' - QuickAssist accelerated DEFLATE*
                                        algorithm with Gzip* extended header.
                          'xpress' - Software based Xpress* algorithm.
                          'igzip' - Software based DEFLATE* algorithm using
                                  igzip DLL.
                          'qatlz4' - QuickAssist accelerated algorithm
                                  with lz4* header.
  -c chunkSizeInKB      Chunk size, in KB.  Default is 64.
                       Files will be divided into chunks of this size
                       (the last chunk may be smaller), and each chunk
                       is compressed separately. State is not maintained
                       between chunks. The chunk size used for decompression
                       must match the value used for compression
  -crc bitWidth         CRC bitwidth. Valid value is 64.
  -pcrc index           CRC64 configuration to use. Valid values are 0 (ECMA-182) or 1␣
↪(Rocksoft). Default is 0.
  -l compressionLevel   Compression level.  Default is 1.
                       compressionLevel can range from 1 - 9. Lower values
                       imply less compressibility in less time.
  -d                    Decompress the input file. Default is to compress.
  -x numLines           Print a summary of the inputs and outputs in a
                       comma-separated variable (CSV) format for easy
                       importing into a spreadsheet. Specify numLines as 1
                       for data only, or 2 to also include a header summary
  -t numThreads         Creates specified number of threads, splits input
                       file into numThreads (near-)equal chunks, and
                       performs the operation in each thread. If specifying
                       multiple threads, -Q is also required.
  -f cpuFreqInMHz       Specifies the CPU frequency in MHz. If not specified,
                       this will be measured (takes approx. 1 second)
  -n numIterations      Specifies the number of iterations (allows you to run
                       the same operation numIterations times) Default is 1.
  -Q                    Test independent threads writing one process using
                       one session. Uses multiple copies of same input file,
                       outputs one output file per thread.
                       Must be used with -t and threadcount of 1 or more.
  -k blockSizeInKB      Separate the source data into several blocks of size
                       specified by blockSizeInKB. -k uses -Q by default.
  -h                    Print this help message.
```

(continues on next page)

(continued from previous page)

```
The following options are applicable for the qat provider only:

   -j maxOutstandingJobs Maximum number of outstanding jobs (requests) that
                         may be outstanding at any one time.  Default is 30.
   -s                     Static compression. Default is dynamic.
   -D                     Dynamic compression. This is default.
   -FB                    Enable igzip fallback.
   -FT thresSizeInKB      Threshold value for fallback. If the offload size
                         is less than the threshold, software provider is used.
```

3.2 Parcomp Examples

Below are a some examples of the results obtained by running Parcomp for compression and decompression.

Note: These examples are for illustrative purposes only.

3.2.1 Compress and Decompress via 'qat'

The following example will compress and decompress a file with the 'qat' provider, which is equivalent of using the Deflate algorithm with data format QZ_DEFLATE_4B. The default compression level is *1* and the default chunk size (QATzip hw_buff_sz) is *64KiB*.

```
>> parcomp.exe -p qat -i largetext -o largetext.compressed

Warning: The hw_buff_sz parameter value used for decompression must match the value used␣
↪for compression.
Default hw_buff_sz value: 65536 bytes.
hw_buff_sz value used in current execution: 65536 bytes.
Parcomp: Tool to test compression & decompression
(c) 2020, Intel(R) Corporation

Reading input file: C:\CompressionFiles\largetext (1000000000 Bytes)
Writing output file: C:\CompressionFiles\largetext.compressed (489318805 Bytes)

Deflation Ratio (%age)     : 48.9
Thruput (uncompressed Mbps): 12312.576
Time (ms)                  : 571.054

Note:- All times exclude file I/O and are measured around the call to the qzCompress()␣
↪API only.

>> parcomp.exe -p qat -d -i largetext.compressed -o largetext.original

Warning: The hw_buff_sz parameter value used for decompression must match the value used␣
↪for compression.
Default hw_buff_sz value: 65536 bytes.
hw_buff_sz value used in current execution: 65536 bytes.
```

(continues on next page)

(continued from previous page)

```
Parcomp: Tool to test compression & decompression
(c) 2020, Intel(R) Corporation

Reading input file: C:\CompressionFiles\largetext.compressed (489318806 Bytes)
Writing output file: C:\CompressionFiles\largetext.original (1000000000 Bytes)

Inflation Ratio (%age)     : 204.4
Thruput (uncompressed Mbps): 23407.884
Time (ms)                  : 280.976

Note:- All times exclude file I/O and are measured around the call to the qzDecompress()␣
↪API only.
```

3.2.2 Compress/Decompress Using Gzip Extended Header

The following example will compress and decompress a file with the 'qatgzipext' provider, which is equivalent of using the Deflate algorithm with data format QZ_DEFLATE_GZIP_EXT. The default compression level is *1* and the default chunk size (QATzip hw_buff_sz) is *64KiB*.

```
>> parcomp.exe -p qatgzipext -i largetext -o largetext.compressed

Warning: The hw_buff_sz parameter value used for decompression must match the value used␣
↪for compression.
Default hw_buff_sz value: 65536 bytes.
hw_buff_sz value used in current execution: 65536 bytes.
Parcomp: Tool to test compression & decompression
(c) 2020, Intel(R) Corporation

Reading input file: largetext.txt (1000000000 Bytes)
Writing output file: largetext.compressed (396497299 Bytes)

Deflation Ratio (%age)     : 39.6
Thruput (uncompressed Mbps): 14677.658
Time (ms)                  : 356.475

Note:- All times exclude file I/O and are measured around the call to the qzCompress()␣
↪API only.

>> parcomp.exe -p qatgzipext -d -i largetext.compressed -o largetext.original

Warning: The hw_buff_sz parameter value used for decompression must match the value used␣
↪for compression.
Default hw_buff_sz value: 65536 bytes.
hw_buff_sz value used in current execution: 65536 bytes.
Parcomp: Tool to test compression & decompression
(c) 2020, Intel(R) Corporation

Reading input file: largetext.compressed (396497299 Bytes)
Writing output file: largetext.original (1000000000 Bytes)

Inflation Ratio (%age)     : 252.2
```

(continues on next page)

(continued from previous page)

```
Thruput (uncompressed Mbps): 24304.489
Time (ms)                  : 188.323

Note:- All times exclude file I/O and are measured around the call to the qzDecompress()␣
↪API only.
```

3.2.3 Compress/Decompress Using Xpress*

The following example will compress and decompress a file with the 'xpress' provider, which is a software based Microsoft compression algorithm.

```
>> parcomp.exe -p xpress -i largetext -o largetext.compressed
Parcomp: Tool to test compression & decompression
(c) 2020, Intel(R) Corporation

Reading input file: C:\CompressionFiles\largetext (1000000000 Bytes)
Writing output file: C:\CompressionFiles\largetext.compressed (573799425 Bytes)

Deflation Ratio (%age)     : 57.4
Thruput (uncompressed Mbps): 1070.143
Time (ms)                  : 7283.892

Note:- All times exclude file I/O and are measured around the call to the qzCompress()␣
↪API only.

$parcomp.exe -p xpress -d -i largetext.compressed -o largetext.original
Parcomp: Tool to test compression & decompression
(c) 2020, Intel(R) Corporation

Reading input file: C:\CompressionFiles\largetext.compressed (573799425 Bytes)
Writing output file: C:\CompressionFiles\largetext.original (1000000000 Bytes)

Inflation Ratio (%age)     : 174.3
Thruput (uncompressed Mbps): 3129.744
Time (ms)                  : 2408.823

Note:- All times exclude file I/O and are measured around the call to the qzDecompress()␣
↪API only.
```

3.2.4 Compress/Decompress Performance

To achieve optimal performance with QATzip via parcomp, it is recommended to use multiple independent threads for compression and/or decompression. This can be achieved using *-t* (multiple threads) and *-Q* (independent threads).

Note: When using *-Q* as a parameter in the compression command, this will produce a number of identical output files with an appended numeric feather starting with 0. The use of *-k* implicitly brings the same enablement of the *-Q* option.

Unless the file is renamed, this feather will also be required for the decompression command. See full example below.

Important: These performance numbers are samples and is dependent on the platform configuration.

```
>> parcomp.exe -i calgary -o calgary.compressed -p qatgzipext -j 30 -Q -t 16 -n 4000 -l␣
↪1 -c 64 -D
Parcomp: Tool to test compression & decompression
(c) 2020, Intel(R) Corporation

Reading input file: D:\CompressionFiles\calgary (3251493 Bytes)
Warning: No Block size specified, using block size 3251493 bytes.
All threads completed as Expected.

Deflation Ratio (%age)     : 35.9
Thruput (uncompressed Mbps): 312145.964
Processing Block size      : 3175 KB
Block count                : 1
Time/block (ms)            : 1.333

Note:- All times exclude file I/O and are measured around the call to the
qzCompressExt() API only.
```

CHAPTER

FOUR

CRYPTOGRAPHY TEST APPLICATION

This package comes with a tool called 'cngtest' to test the performance of the Intel® QuickAssist Technology accelerator. Cngtest has been built using with the Microsoft CNG framework found in the Cryptography API: Next Generation Software Development Kit. The cngtest binary can be found in the following folder upon completion of the installation:

Program FilesIntelIntel(R) QuickAssist TechnologyCryptoSamplesbin

4.1 Cngtest Examples

Below are a some examples of the results obtained by running cngtest for cryptography.

Note: These examples are for illustrative purposes only.

4.1.1 RSA 3072-bit Key

The following example will use cngtest to run RSA 3072-bit Key for encrypt and decrypt operations.

```
>> cngtest.exe -provider=qa -algo=rsa -keyLength=3072
----- QAT Device: Intel(R) 4xxx Accelerator
----- Number of Devices: 4
----- Number of enabled Devices: 4
----- Max number of threads: 192

Running in user space...
[RunTest] Running with 2 thread(s).
   Time [ns]                          : 2891517000
   Number of iterations               : 100000
   RSA Encrypt Ops/s                  : 34583.92
   CPU core utilization percentage    : 3%
   CPU overall utilization percentage : 1%
   Time [ns]                          : 50226472900
   Number of iterations               : 100000
   RSA Decrypt Ops/s                  : 1990.98
   CPU core utilization percentage    : 1%
   CPU overall utilization percentage : 0%
```

4.1.2 ECDH using Curve25519

The following example will use cngtest to run ECDH with Curve255190 operations.

```
>> cngtest.exe -provider=qa -algo=ecdh -ecccurve=curve25519
----- QAT Device: Intel(R) 4xxx Accelerator
----- Number of Devices: 4
----- Number of enabled Devices: 4
----- Max number of threads: 192

Running in user space...
[RunTest] Running with 2 thread(s).
   Time [ns]                          : 3873229000
   Number of iterations               : 100000
   ECDH Stage 1 Ops/s                 : 25818.25
   CPU core utilization percentage    : 3%
   CPU overall utilization percentage : 2%
   Time [ns]                          : 2840583800
   Number of iterations               : 100000
   ECDH Stage 2 Ops/s                 : 35204.03
   CPU core utilization percentage    : 2%
   CPU overall utilization percentage : 2%
```

4.1.3 ECDSA using NistP521

The following example will use cngtest to run ECDSA with NistP521 operations.

```
>> .\cngtest.exe -provider=qa -algo=ecdsa -ecccurve=nistP521
----- QAT Device: Intel(R) 4xxx Accelerator
----- Number of Devices: 4
----- Number of enabled Devices: 4
----- Max number of threads: 192

Running in user space...
[RunTest] Running with 2 thread(s).
   Time [ns]                          : 20686033200
   Number of iterations               : 100000
   ECDSA Sign Hash Ops/s              : 4834.18
   CPU core utilization percentage    : 1%
   CPU overall utilization percentage : 0%
   Time [ns]                          : 68163263500
   Number of iterations               : 100000
   ECDSA Verify Signature Ops/s       : 1467.07
   CPU core utilization percentage    : 1%
   CPU overall utilization percentage : 0%
```

4.1.4 RSA 2048-bit Key Performance

To achieve optimal performance with QAT hardware offloading using Microsoft* CNG framework, it is recommended to use multiple threads for cryptography operations with a set affinity mask.

Important: These performance numbers are samples and is dependent on the platform configuration.

```
>> cngtest.exe -provider=qa -algo=rsa -keyLength=2048 -numThreads=96 -affinityMask=0xff
----- QAT Device: Intel(R) 4xxx Accelerator
----- Number of Devices: 4
----- Number of enabled Devices: 4
----- Max number of threads: 192

Running in user space...
[RunTest] Running with 96 thread(s).
   Time [ns]                          : 294058600
   Number of iterations               : 100000
   RSA Encrypt Ops/s                  : 340068.27
   CPU core utilization percentage    : 85%
   CPU overall utilization percentage : 13%
   Time [ns]                          : 1264662500
   Number of iterations               : 100000
   RSA Decrypt Ops/s                  : 79072.48
   CPU core utilization percentage    : 12%
   CPU overall utilization percentage : 3%
```