

QATzip  
1.0.2

Generated by Doxygen 1.8.5

Fri Nov 20 2020 03:25:31



# Contents

<b>1</b>	<b>Module Index</b>	<b>1</b>
1.1	Modules . . . . .	1
<b>2</b>	<b>Class Index</b>	<b>3</b>
2.1	Class List . . . . .	3
<b>3</b>	<b>File Index</b>	<b>5</b>
3.1	File List . . . . .	5
<b>4</b>	<b>Module Documentation</b>	<b>7</b>
4.1	Data Compression API . . . . .	7
4.1.1	Detailed Description . . . . .	8
4.1.2	Macro Definition Documentation . . . . .	8
4.1.2.1	QATZIP_API_VERSION_NUM_MAJOR . . . . .	8
4.1.2.2	QATZIP_API_VERSION_NUM_MINOR . . . . .	8
4.1.2.3	QZ_OK . . . . .	8
4.1.2.4	QZ_SKID_PAD_SZ . . . . .	9
4.1.3	Typedef Documentation . . . . .	9
4.1.3.1	PinMem_T . . . . .	9
4.1.3.2	QzCrcType_T . . . . .	9
4.1.3.3	QzDataFormat_T . . . . .	9
4.1.3.4	QzDirection_T . . . . .	10
4.1.3.5	QzHuffmanHdr_T . . . . .	10
4.1.3.6	QzSession_T . . . . .	10
4.1.3.7	QzSessionParams_T . . . . .	10
4.1.3.8	QzStatus_T . . . . .	11
4.1.3.9	QzStream_T . . . . .	11
4.1.4	Enumeration Type Documentation . . . . .	11
4.1.4.1	PinMem_E . . . . .	11
4.1.4.2	QzCrcType_E . . . . .	11
4.1.4.3	QzDataFormat_E . . . . .	11
4.1.4.4	QzDirection_E . . . . .	12

4.1.4.5	QzHuffmanHdr_E	12
4.1.5	Function Documentation	13
4.1.5.1	qzClose	13
4.1.5.2	qzCompress	13
4.1.5.3	qzCompressCrc	14
4.1.5.4	qzCompressStream	15
4.1.5.5	qzDecompress	17
4.1.5.6	qzDecompressStream	18
4.1.5.7	qzEndStream	19
4.1.5.8	qzFree	20
4.1.5.9	qzGetDefaults	21
4.1.5.10	qzGetStatus	22
4.1.5.11	qzInit	22
4.1.5.12	qzMalloc	23
4.1.5.13	qzMemFindAddr	24
4.1.5.14	qzSetDefaults	24
4.1.5.15	qzSetupSession	25
4.1.5.16	qzTeardownSession	26
<b>5</b>	<b>Class Documentation</b>	<b>27</b>
5.1	QzSession_S Struct Reference	27
5.1.1	Detailed Description	27
5.1.2	Member Data Documentation	27
5.1.2.1	hw_session_stat	27
5.1.2.2	internal	27
5.1.2.3	thd_sess_stat	27
5.1.2.4	total_in	27
5.1.2.5	total_out	28
5.2	QzSessionParams_S Struct Reference	28
5.2.1	Detailed Description	28
5.2.2	Member Data Documentation	28
5.2.2.1	comp_algorithm	28
5.2.2.2	comp_lvl	28
5.2.2.3	data_fmt	28
5.2.2.4	direction	28
5.2.2.5	huffman_hdr	29
5.2.2.6	hw_buff_sz	29
5.2.2.7	input_sz_thrshold	29
5.2.2.8	max_forks	29
5.2.2.9	req_cnt_thrshold	29

5.2.2.10	strm_buff_sz	29
5.2.2.11	sw_backup	29
5.2.2.12	wait_cnt_thrshold	29
5.3	QzStatus_S Struct Reference	29
5.3.1	Detailed Description	30
5.3.2	Member Data Documentation	30
5.3.2.1	algo_hw	30
5.3.2.2	algo_sw	30
5.3.2.3	hw_session_status	30
5.3.2.4	memory_allocated	30
5.3.2.5	qat_hw_count	30
5.3.2.6	qat_instance_attach	30
5.3.2.7	qat_mem_drvr	30
5.3.2.8	qat_service_stated	30
5.3.2.9	using_huge_pages	30
5.4	QzStream_S Struct Reference	31
5.4.1	Detailed Description	31
5.4.2	Member Data Documentation	31
5.4.2.1	crc_32	31
5.4.2.2	crc_type	31
5.4.2.3	in	31
5.4.2.4	in_sz	31
5.4.2.5	opaque	31
5.4.2.6	out	31
5.4.2.7	out_sz	32
5.4.2.8	pending_in	32
5.4.2.9	pending_out	32
5.4.2.10	reserved	32
<b>6</b>	<b>File Documentation</b>	<b>33</b>
6.1	include/qatzip.h File Reference	33
6.1.1	Macro Definition Documentation	35
6.1.1.1	MIN	35
6.1.1.2	QATZIP_API	35
6.1.1.3	QATZIP_API_VERSION	35
6.1.1.4	QZ_BUF_ERROR	35
6.1.1.5	QZ_COMP_ALGOL_DEFAULT	35
6.1.1.6	QZ_COMP_LEVEL_DEFAULT	35
6.1.1.7	QZ_COMP_THRESHOLD_DEFAULT	35
6.1.1.8	QZ_COMP_THRESHOLD_MINIMUM	35

6.1.1.9	QZ_COMPRESSED_SZ_OF_EMPTY_FILE	36
6.1.1.10	QZ_DATA_ERROR	36
6.1.1.11	QZ_DATA_FORMAT_DEFAULT	36
6.1.1.12	QZ_DEFLATE	36
6.1.1.13	QZ_DEFLATE_COMP_LVL_MAXIMUM	36
6.1.1.14	QZ_DEFLATE_COMP_LVL_MINIMUM	36
6.1.1.15	QZ_DIRECTION_DEFAULT	36
6.1.1.16	QZ_DUPLICATE	36
6.1.1.17	QZ_FAIL	36
6.1.1.18	QZ_FORCE_SW	36
6.1.1.19	QZ_HUFF_HDR_DEFAULT	36
6.1.1.20	QZ_HW_BUFF_MAX_SZ	36
6.1.1.21	QZ_HW_BUFF_MIN_SZ	36
6.1.1.22	QZ_HW_BUFF_SZ	36
6.1.1.23	QZ_LOW_DEST_MEM	36
6.1.1.24	QZ_LOW_MEM	36
6.1.1.25	QZ_MAX_ALGORITHMS	36
6.1.1.26	QZ_MAX_FORK_DEFAULT	36
6.1.1.27	QZ_MEMCPY	36
6.1.1.28	QZ_NO_HW	37
6.1.1.29	QZ_NO_INST_ATTACH	37
6.1.1.30	QZ_NO_MDRV	37
6.1.1.31	QZ_NONE	37
6.1.1.32	QZ_NOSW_LOW_MEM	37
6.1.1.33	QZ_NOSW_NO_HW	37
6.1.1.34	QZ_NOSW_NO_INST_ATTACH	37
6.1.1.35	QZ_NOSW_NO_MDRV	37
6.1.1.36	QZ_PARAMS	37
6.1.1.37	QZ_POLL_SLEEP_DEFAULT	37
6.1.1.38	QZ_REQ_THRESHOLD_DEFAULT	37
6.1.1.39	QZ_REQ_THRESHOLD_MAXIMUM	37
6.1.1.40	QZ_REQ_THRESHOLD_MINIMUM	37
6.1.1.41	QZ_STRM_BUFF_MAX_SZ	37
6.1.1.42	QZ_STRM_BUFF_MIN_SZ	37
6.1.1.43	QZ_STRM_BUFF_SZ_DEFAULT	37
6.1.1.44	QZ_SW_BACKUP_DEFAULT	38
6.1.1.45	QZ_WAIT_CNT_THRESHOLD_DEFAULT	38
6.1.2	Function Documentation	38
6.1.2.1	qzMaxCompressedLength	38

**Index****39**





# Chapter 1

## Module Index

### 1.1 Modules

Here is a list of all modules:

Data Compression API . . . . .	7
--------------------------------	---



## Chapter 2

# Class Index

### 2.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

<a href="#">QzSession_S</a> . . . . .	27
<a href="#">QzSessionParams_S</a> . . . . .	28
<a href="#">QzStatus_S</a> . . . . .	29
<a href="#">QzStream_S</a> . . . . .	31



## Chapter 3

# File Index

### 3.1 File List

Here is a list of all files with brief descriptions:

include/ <a href="#">qatzip.h</a> . . . . .	33
---	----



## Chapter 4

# Module Documentation

### 4.1 Data Compression API

#### Classes

- struct [QzSessionParams\\_S](#)
- struct [QzSession\\_S](#)
- struct [QzStatus\\_S](#)
- struct [QzStream\\_S](#)

#### Macros

- #define [QATZIP\\_API\\_VERSION\\_NUM\\_MAJOR](#) (1)
- #define [QATZIP\\_API\\_VERSION\\_NUM\\_MINOR](#) (2)
- #define [QZ\\_OK](#) (0)
- #define [QZ\\_SKID\\_PAD\\_SZ](#) 48

#### Typedefs

- typedef enum [QzHuffmanHdr\\_E](#) [QzHuffmanHdr\\_T](#)
- typedef enum [PinMem\\_E](#) [PinMem\\_T](#)
- typedef enum [QzDirection\\_E](#) [QzDirection\\_T](#)
- typedef enum [QzDataFormat\\_E](#) [QzDataFormat\\_T](#)
- typedef enum [QzCrcType\\_E](#) [QzCrcType\\_T](#)
- typedef struct [QzSessionParams\\_S](#) [QzSessionParams\\_T](#)
- typedef struct [QzSession\\_S](#) [QzSession\\_T](#)
- typedef struct [QzStatus\\_S](#) [QzStatus\\_T](#)
- typedef struct [QzStream\\_S](#) [QzStream\\_T](#)

#### Enumerations

- enum [QzHuffmanHdr\\_E](#) { [QZ\\_DYNAMIC\\_HDR](#) = 0, [QZ\\_STATIC\\_HDR](#) }
- enum [PinMem\\_E](#) { [COMMON\\_MEM](#) = 0, [PINNED\\_MEM](#) }
- enum [QzDirection\\_E](#) { [QZ\\_DIR\\_COMPRESS](#) = 0, [QZ\\_DIR\\_DECOMPRESS](#), [QZ\\_DIR\\_BOTH](#) }
- enum [QzDataFormat\\_E](#) {  
    [QZ\\_DEFLATE\\_4B](#) = 0, [QZ\\_DEFLATE\\_GZIP](#), [QZ\\_DEFLATE\\_GZIP\\_EXT](#), [QZ\\_DEFLATE\\_RAW](#),  
    [QZ\\_FMT\\_NUM](#) }
- enum [QzCrcType\\_E](#) { [QZ\\_CRC32](#) = 0, [QZ\\_ADLER](#), [NONE](#) }

## Functions

- [QATZIP\\_API](#) int [qzInit](#) ([QzSession\\_T](#) \*sess, unsigned char sw\_backup)
- [QATZIP\\_API](#) int [qzSetupSession](#) ([QzSession\\_T](#) \*sess, [QzSessionParams\\_T](#) \*params)
- [QATZIP\\_API](#) int [qzCompress](#) ([QzSession\\_T](#) \*sess, const unsigned char \*src, unsigned int \*src\_len, unsigned char \*dest, unsigned int \*dest\_len, unsigned int last)
- [QATZIP\\_API](#) int [qzCompressCrc](#) ([QzSession\\_T](#) \*sess, const unsigned char \*src, unsigned int \*src\_len, unsigned char \*dest, unsigned int \*dest\_len, unsigned int last, unsigned long \*crc)
- [QATZIP\\_API](#) int [qzDecompress](#) ([QzSession\\_T](#) \*sess, const unsigned char \*src, unsigned int \*src\_len, unsigned char \*dest, unsigned int \*dest\_len)
- [QATZIP\\_API](#) int [qzTeardownSession](#) ([QzSession\\_T](#) \*sess)
- [QATZIP\\_API](#) int [qzClose](#) ([QzSession\\_T](#) \*sess)
- [QATZIP\\_API](#) int [qzGetStatus](#) ([QzSession\\_T](#) \*sess, [QzStatus\\_T](#) \*status)
- [QATZIP\\_API](#) int [qzSetDefaults](#) ([QzSessionParams\\_T](#) \*defaults)
- [QATZIP\\_API](#) int [qzGetDefaults](#) ([QzSessionParams\\_T](#) \*defaults)
- [QATZIP\\_API](#) void \* [qzMalloc](#) (size\_t sz, int numa, int force\_pinned)
- [QATZIP\\_API](#) void [qzFree](#) (void \*m)
- [QATZIP\\_API](#) int [qzMemFindAddr](#) (unsigned char \*a)
- [QATZIP\\_API](#) int [qzCompressStream](#) ([QzSession\\_T](#) \*sess, [QzStream\\_T](#) \*strm, unsigned int last)
- [QATZIP\\_API](#) int [qzDecompressStream](#) ([QzSession\\_T](#) \*sess, [QzStream\\_T](#) \*strm, unsigned int last)
- [QATZIP\\_API](#) int [qzEndStream](#) ([QzSession\\_T](#) \*sess, [QzStream\\_T](#) \*strm)

### 4.1.1 Detailed Description

These functions specify the API for Data Compression operations.

Remarks

### 4.1.2 Macro Definition Documentation

#### 4.1.2.1 #define QATZIP\_API\_VERSION\_NUM\_MAJOR (1)

`QATZIP Major Version Number`

The QATZIP API major version number. This number will be incremented when significant changes to the API has occurred. The combination of the major and minor number definitions represent the complete version number for this interface.

#### 4.1.2.2 #define QATZIP\_API\_VERSION\_NUM\_MINOR (2)

`QATZIP Minor Version Number`

The QATZIP API minor version number. This number will be incremented when minor changes to the API has occurred. The combination of the major and minor number definitions represent the complete version number for this interface.

#### 4.1.2.3 #define QZ\_OK (0)

`QATZIP Session Status definitions and function return codes`

This list identifies valid values for session status and function return codes. Success



4.1.2.4 `#define QZ_SKID_PAD_SZ 48`

Get the max compressed output length

Get the max compressed output length.

This function shall not be called in an interrupt context. None None Yes No Yes

## Parameters

<code>in</code>	<code>src_sz</code>	Input data length in byte sess Session handle (pointer to opaque instance and session data)
-----------------	---------------------	---

## Return values

<code>dest_sz</code>	Max compressed data output length in byte. When <code>src_sz</code> is equal to 0, the return value is <a href="#">QZ_COMPRESSED_SZ_OF_EMPTY_FILE(34)</a> . When integer overflow happens, the return value is 0
----------------------	--

## Precondition

None

## Postcondition

None

## Note

Only a synchronous version of this function is provided.

## See Also

None

## 4.1.3 Typedef Documentation

4.1.3.1 `typedef enum PinMem_E PinMem_T`

Supported memory types

This enumerated list identifies memory types supported by QATZip.

4.1.3.2 `typedef enum QzCrcType_E QzCrcType_T`

Supported checksum type

This enumerated list identifies the checksum type for input/output data. The format can be CRC32, Adler or none.

4.1.3.3 `typedef enum QzDataFormat_E QzDataFormat_T`

Streaming API input and output format

This enumerated list identifies the data format supported by QATZip streaming API. A format can be raw deflate data block, deflate block wrapped by GZip header and footer, or deflate data block wrapped by GZip extension header and footer.

#### 4.1.3.4 typedef enum QzDirection\_E QzDirection\_T

Compress or decompress setting

This enumerated list identifies the session directions supported by QATZip. A session can be compress, decompress or both.

#### 4.1.3.5 typedef enum QzHuffmanHdr\_E QzHuffmanHdr\_T

This API provides access to underlying compression functions in QAT hardware. The API supports an implementation that provides compression service in software if not all of the required resources are available to execute the compression service in hardware.

The API supports threaded applications. Applications can create threads and each of these threads can invoke the API defined herein.

For simplicity, initializations and setup function calls are not required to obtain compression services. If the initialization and setup functions are not called before compression or decompression requests, then they will be called with default arguments from within the compression or decompression functions. This results in several legal calling scenarios, described below.

Scenario 1 - All functions explicitly invoked by caller, with all arguments provided.

```
qzInit(&sess, sw_backup); qzSetupSession(&sess, &params); qzCompress(&sess, src, &src_len, dest, &dest_len, 1); qzDecompress(&sess, src, &src_len, dest, &dest_len); qzTeardownSession(&sess); qzClose(&sess);
```

Scenario 2 - Initialization function called, setup function not invoked by caller. This scenario can be used to specify the sw\_backup argument to qzInit.

```
qzInit(&sess, sw_backup); qzCompress(&sess, src, &src_len, dest, &dest_len, 1); calls qzSetupSession(sess, NULL); qzTeardownSession(&sess); qzClose(&sess);
```

Scenario 3 - Calling application simply invokes the actual qzCompress functions.

```
qzCompress(&sess, src, &src_len, dest, &dest_len, 0); calls qzInit(sess, 1); calls qzSetupSession(sess, NULL); qzCompress(&sess, src, &src_len, dest, &dest_len, 1);
```

Notes: Invoking qzSetupSession with NULL for params sets up a session with default session attributed, detailed in the function description below.

If an application terminates with out invoking tear down and close functions, process termination will invoke memory and hardware instance cleanup.

If a thread terminates without invoking tear down and close functions, memory and hardware are not cleaned up until the application exits.

Supported Huffman Headers

This enumerated list identifies the Huffman header types supported by QATZip.

#### 4.1.3.6 typedef struct QzSession\_S QzSession\_T

QATZIP Session opaque data storage

This structure contains a pointer to a structure with session state.

#### 4.1.3.7 typedef struct QzSessionParams\_S QzSessionParams\_T

QATZIP Session Initialization parameters

This structure contains data for initializing a session.

## 4.1.3.8 typedef struct QzStatus\_S QzStatus\_T

QATZIP status structure

This structure contains data relating to the status of QAT on the platform.

## 4.1.3.9 typedef struct QzStream\_S QzStream\_T

QATZIP Stream data storage

This structure contains metadata needed for stream operation.

## 4.1.4 Enumeration Type Documentation

## 4.1.4.1 enum PinMem\_E

Supported memory types

This enumerated list identifies memory types supported by QATZip.

## Enumerator

**COMMON\_MEM** Allocate non-contiguous memory

**PINNED\_MEM** Allocate contiguous memory

## 4.1.4.2 enum QzCrcType\_E

Supported checksum type

This enumerated list identifies the checksum type for input/output data. The format can be CRC32, Adler or none.

## Enumerator

**QZ\_CRC32** CRC32 checksum

**QZ\_ADLER** Adler checksum

**NONE** No checksum

## 4.1.4.3 enum QzDataFormat\_E

Streaming API input and output format

This enumerated list identifies the data format supported by QATZip streaming API. A format can be raw deflate data block, deflate block wrapped by GZip header and footer, or deflate data block wrapped by GZip extension header and footer.

## Enumerator

**QZ\_DEFLATE\_4B** Data is in raw deflate format with 4 byte header

**QZ\_DEFLATE\_GZIP** Data is in deflate wrapped by GZip header and footer

**QZ\_DEFLATE\_GZIP\_EXT** Data is in deflate wrapped by GZip extension header and footer

**QZ\_DEFLATE\_RAW** Data is in raw deflate format

**QZ\_FMT\_NUM**

#### 4.1.4.4 enum QzDirection\_E

Compress or decompress setting

This enumerated list identifies the session directions supported by QATZip. A session can be compress, decompress or both.

##### Enumerator

**QZ\_DIR\_COMPRESS** Session will be used for compression

**QZ\_DIR\_DECOMPRESS** Session will be used for decompression

**QZ\_DIR\_BOTH** Session will be used for both compression and decompression

#### 4.1.4.5 enum QzHuffmanHdr\_E

This API provides access to underlying compression functions in QAT hardware. The API supports an implementation that provides compression service in software if not all of the required resources are available to execute the compression service in hardware.

The API supports threaded applications. Applications can create threads and each of these threads can invoke the API defined herein.

For simplicity, initializations and setup function calls are not required to obtain compression services. If the initialization and setup functions are not called before compression or decompression requests, then they will be called with default arguments from within the compression or decompression functions. This results in several legal calling scenarios, described below.

Scenario 1 - All functions explicitly invoked by caller, with all arguments provided.

```
qzInit(&sess, sw_backup); qzSetupSession(&sess, &params); qzCompress(&sess, src, &src_len, dest, &dest_len, 1); qzDecompress(&sess, src, &src_len, dest, &dest_len); qzTeardownSession(&sess); qzClose(&sess);
```

Scenario 2 - Initialization function called, setup function not invoked by caller. This scenario can be used to specify the sw\_backup argument to qzInit.

```
qzInit(&sess, sw_backup); qzCompress(&sess, src, &src_len, dest, &dest_len, 1); calls qzSetupSession(sess, NULL); qzTeardownSession(&sess); qzClose(&sess);
```

Scenario 3 - Calling application simply invokes the actual qzCompress functions.

```
qzCompress(&sess, src, &src_len, dest, &dest_len, 0); calls qzInit(sess, 1); calls qzSetupSession(sess, NULL); qzCompress(&sess, src, &src_len, dest, &dest_len, 1);
```

Notes: Invoking qzSetupSession with NULL for params sets up a session with default session attributed, detailed in the function description below.

If an application terminates with out invoking tear down and close functions, process termination will invoke memory and hardware instance cleanup.

If a thread terminates without invoking tear down and close functions, memory and hardware are not cleaned up until the application exits.

Supported Huffman Headers

This enumerated list identifies the Huffman header types supported by QATZip.

##### Enumerator

**QZ\_DYNAMIC\_HDR** Full Dynamic Huffman Trees

**QZ\_STATIC\_HDR** Static Huffman Trees

### 4.1.5 Function Documentation

#### 4.1.5.1 QATZIP\_API int qzClose ( QzSession\_T \* sess )

Terminates a QATzip session

This function closes the connection with QAT.

This function shall not be called in an interrupt context None None Yes No Yes

##### Parameters

in	sess	Session handle (pointer to opaque instance and session data)
----	------	--

##### Return values

QZ_OK	Function executed successfully
QZ_FAIL	Function did not succeed
QZ_PARAMS	*sess is NULL or member of params is invalid

##### Precondition

None

##### Postcondition

None

##### Note

Only a synchronous version of this function is provided.

##### See Also

None

#### 4.1.5.2 QATZIP\_API int qzCompress ( QzSession\_T \* sess, const unsigned char \* src, unsigned int \* src\_len, unsigned char \* dest, unsigned int \* dest\_len, unsigned int last )

Compress a buffer

This function will compress a buffer if either a hardware based session or a software based session is available. If no session has been established - as indicated by the contents of \*sess - then this function will attempt to set up a session using qzInit and qzSetupSession.

The resulting compressed block of data will be composed of one or more gzip blocks per RFC 1952.

This function will place completed compression blocks in the output buffer.

The caller must check the updated src\_len. This value will be the number of consumed bytes on exit. The calling API may have to process the destination buffer and call again.

The parameter dest\_len will be set to the number of bytes produced in the destination buffer. This value may be zero if no data was produced which may occur if the consumed data is retained internally. A possible reason for this may be small amounts of data in the src buffer.

This function shall not be called in an interrupt context. None None Yes No Yes

**Parameters**

in	<i>sess</i>	Session handle (pointer to opaque instance and session data)
in	<i>src</i>	Point to source buffer
in, out	<i>src_len</i>	Length of source buffer. Modified to number of bytes consumed
in	<i>dest</i>	Point to destination buffer
in, out	<i>dest_len</i>	Length of destination buffer. Modified to length of compressed data when function returns
in	<i>last</i>	1 for 'No more data to be compressed' 0 for 'More data to be compressed'

**Return values**

<i>QZ_OK</i>	Function executed successfully
<i>QZ_FAIL</i>	Function did not succeed
<i>QZ_PARAMS</i>	*sess is NULL or member of params is invalid

**Precondition**

None

**Postcondition**

None

**Note**

Only a synchronous version of this function is provided.

**See Also**

None

#### 4.1.5.3 QATZIP\_API int qzCompressCrc ( QzSession\_T \* sess, const unsigned char \* src, unsigned int \* src\_len, unsigned char \* dest, unsigned int \* dest\_len, unsigned int last, unsigned long \* crc )

Compress a buffer and return the CRC checksum

This function will compress a buffer if either a hardware based session or a software based session is available. If no session has been established - as indicated by the contents of \*sess - then this function will attempt to set up a session using qzInit and qzSetupSession.

The resulting compressed block of data will be composed of one or more gzip blocks per RFC 1952.

This function will place completed compression blocks in the output buffer and put CRC32 checksum for compressed input data in user provided buffer \*crc.

The caller must check the updated src\_len. This value will be the number of consumed bytes on exit. The calling API may have to process the destination buffer and call again.

The parameter dest\_len will be set to the number of bytes produced in the destination buffer. This value may be zero if no data was produced which may occur if the consumed data is retained internally. A possible reason for this may be small amounts of data in the src buffer.

This function shall not be called in an interrupt context. None None Yes No Yes

**Parameters**

in	<i>sess</i>	Session handle (pointer to opaque instance and session data)
in	<i>src</i>	Point to source buffer
in, out	<i>src_len</i>	Length of source buffer. Modified to number of bytes consumed
in	<i>dest</i>	Point to destination buffer
in, out	<i>dest_len</i>	Length of destination buffer. Modified to length of compressed data when function returns
in	<i>last</i>	1 for 'No more data to be compressed' 0 for 'More data to be compressed'
in, out	<i>crc</i>	Point to CRC32 checksum buffer

**Return values**

<i>QZ_OK</i>	Function executed successfully
<i>QZ_FAIL</i>	Function did not succeed
<i>QZ_PARAMS</i>	*sess is NULL or member of params is invalid

**Precondition**

None

**Postcondition**

None

**Note**

Only a synchronous version of this function is provided.

**See Also**

None

**4.1.5.4 QATZIP\_API int qzCompressStream ( QzSession\_T \* sess, QzStream\_T \* strm, unsigned int last )**

Compress data in stream and return checksum

This function will compress data in stream buffer if either a hardware based session or a software based session is available. If no session has been established - as indicated by the contents of \*sess - then this function will attempt to set up a session using qzInit and qzSetupSession. The function will start to compress the data when receiving sufficient number of bytes - as defined by hw\_buff\_sz in QzSessionParams\_T - or reaching the end of input data - as indicated by last parameter.

The resulting compressed block of data will be composed of one or more gzip blocks per RFC 1952 or deflate blocks per RFC 1951.

This function will place completed compression blocks in the \*out of QzStream\_T structure and put checksum for compressed input data in crc32 of QzStream\_T structure.

The caller must check the updated in\_sz of QzStream\_T. This value will be the number of consumed bytes on exit. The calling API may have to process the destination buffer and call again.

The parameter out\_sz in QzStream\_T will be set to the number of bytes produced in the destination buffer. This value may be zero if no data was produced which may occur if the consumed data is retained internally. A possible reason for this may be small amounts of data in the src buffer.

The caller must check the updated pending\_in of QzStream\_T. This value will be the number of unprocessed bytes held in QATzip. The calling API may have to feed more input data or indicate reaching the end of input and call again.

The caller must check the updated `pending_out` of `QzStream_T`. This value will be the number of processed bytes held in `QATZip`. The calling API may have to process the destination buffer and call again.

This function shall not be called in an interrupt context. None None Yes No Yes



**Parameters**

<i>in</i>	<i>sess</i>	Session handle (pointer to opaque instance and session data)
<i>in, out</i>	<i>strm</i>	Stream handle
<i>in</i>	<i>last</i>	1 for 'No more data to be compressed' 0 for 'More data to be compressed'

**Return values**

<i>QZ_OK</i>	Function executed successfully
<i>QZ_FAIL</i>	Function did not succeed
<i>QZ_PARAMS</i>	*sess is NULL or member of params is invalid

**Precondition**

None

**Postcondition**

None

**Note**

Only a synchronous version of this function is provided.

**See Also**

None

#### 4.1.5.5 QATZIP\_API int qzDecompress ( QzSession\_T \* sess, const unsigned char \* src, unsigned int \* src\_len, unsigned char \* dest, unsigned int \* dest\_len )

Decompress a buffer

This function will decompress a buffer if either a hardware based session or a software based session is available. If no session has been established - as indicated by the contents of \*sess - then this function will attempt to set up a session using qzInit and qzSetupSession.

The input compressed block of data will be composed of one or more gzip blocks per RFC 1952.

This function shall not be called in an interrupt context. None None Yes No Yes

**Parameters**

<i>in</i>	<i>sess</i>	Session handle (pointer to opaque instance and session data)
<i>in</i>	<i>src</i>	Point to source buffer
<i>in</i>	<i>src_len</i>	Length of source buffer. Modified to length of processed compressed data when function returns
<i>in</i>	<i>dest</i>	Point to destination buffer
<i>in, out</i>	<i>dest_len</i>	Length of destination buffer. Modified to length of decompressed data when function returns

**Return values**

<i>QZ_OK</i>	Function executed successfully
--------------	--------------------------------

<i>QZ_FAIL</i>	Function did not succeed
<i>QZ_PARAMS</i>	*sess is NULL or member of params is invalid

**Precondition**

None

**Postcondition**

None

**Note**

Only a synchronous version of this function is provided.

**See Also**

None

#### 4.1.5.6 QATZIP\_API int qzDecompressStream ( QzSession\_T \* sess, QzStream\_T \* strm, unsigned int last )

Decompress data in stream and return checksum

This function will decompress data in stream buffer if either a hardware based session or a software based session is available. If no session has been established - as indicated by the contents of \*sess - then this function will attempt to set up a session using qzInit and qzSetupSession. The function will start to decompress the data when receiving sufficient number of bytes - as defined by hw\_buff\_sz in QzSessionParams\_T - or reaching the end of input data - as indicated by last parameter.

The input compressed block of data will be composed of one or more gzip blocks per RFC 1952 or deflate blocks per RFC 1951.

This function will place completed decompression blocks in the \*out of QzStream\_T structure and put checksum for decompressed data in crc32 of QzStream\_T structure.

The caller must check the updated in\_sz of QzStream\_T. This value will be the number of consumed bytes on exit. The calling API may have to process the destination buffer and call again.

The parameter out\_sz in QzStream\_T will be set to the number of bytes produced in the destination buffer. This value may be zero if no data was produced which may occur if the consumed data is retained internally. A possible reason for this may be small amounts of data in the src buffer.

The caller must check the updated pending\_in of QzStream\_T. This value will be the number of unprocessed bytes held in QATZip. The calling API may have to feed more input data or indicate reaching the end of input and call again.

The caller must check the updated pending\_out of QzStream\_T. This value will be the number of processed bytes held in QATZip. The calling API may have to process the destination buffer and call again.

This function shall not be called in an interrupt context. None None Yes No Yes

**Parameters**

in	sess	Session handle (pointer to opaque instance and session data)
in, out	strm	Stream handle
in	last	1 for 'No more data to be compressed' 0 for 'More data to be compressed'

## Return values

<i>QZ_OK</i>	Function executed successfully
<i>QZ_FAIL</i>	Function did not succeed
<i>QZ_PARAMS</i>	*sess is NULL or member of params is invalid
<i>QZ_NEED_MORE</i>	*last is set but end of block is absent

## Precondition

None

## Postcondition

None

## Note

Only a synchronous version of this function is provided.

## See Also

None

## 4.1.5.7 QATZIP\_API int qzEndStream ( QzSession\_T \* sess, QzStream\_T \* strm )

Terminates a QATZip stream

This function disconnect stream handle from session handle then reset stream flag and release stream memory.

This function shall not be called in an interrupt context. None None Yes No Yes

## Parameters

in	sess	Session handle (pointer to opaque instance and session data)
----	------	--

## Return values

<i>QZ_OK</i>	Function executed successfully
<i>QZ_FAIL</i>	Function did not succeed
<i>QZ_PARAMS</i>	*sess is NULL or member of params is invalid

## Precondition

None

## Postcondition

None

## Note

Only a synchronous version of this function is provided.

## See Also

None

**4.1.5.8 QATZIP\_API void qzFree ( void \* *m* )**

Free allocated memory

Free allocated memory.

This function shall not be called in an interrupt context. None None Yes No Yes

**Parameters**

<i>in</i>	<i>m</i>	Memory address to be freed
-----------	----------	----------------------------

**Precondition**

None

**Postcondition**

None

**Note**

Only a synchronous version of this function is provided.

**See Also**

None

**4.1.5.9 QATZIP\_API int qzGetDefaults ( QzSessionParams\_T \* defaults )**

Get default QzSessionParams\_T value

Get default QzSessionParams\_T value.

This function shall not be called in an interrupt context. None None Yes No Yes

**Parameters**

<i>in</i>	<i>defaults</i>	The pointer to default value
-----------	-----------------	------------------------------

**Return values**

<i>QZ_OK</i>	Get default value successfully
<i>QZ_PARAM</i>	Fail to get default value

**Precondition**

None

**Postcondition**

None

**Note**

Only a synchronous version of this function is provided.

**See Also**

None

#### 4.1.5.10 QATZIP\_API int qzGetStatus ( QzSession\_T \* sess, QzStatus\_T \* status )

Get current QAT status

This function retrieves the status of QAT in the platform. The status structure will be filled in as follows: qat\_hw\_count Number of discovered QAT devices on PCU bus qat\_service\_stated 1 if qzInit has been successfully run, 0 otherwise qat\_mem\_drvr 1 if the QAT memory driver is installed, 0 otherwise qat\_instance\_attach 1 if session has attached to a hardware instance, 0 otherwise memory\_allocated Amount of memory, in kilobytes, from kernel or huge pages allocated by this process/thread. using\_huge\_pages 1 if memory is being allocated from huge pages, 0 if memory is being allocated from standard kernel memory hw\_session\_stat Hw session status: one of: QZ\_OK QZ\_FAIL QZ\_NO\_HW QZ\_NO\_MDRV QZ\_NO\_INST\_ATTACH QZ\_LOW\_MEM QZ\_NOSW\_NO\_HW QZ\_NOSW\_MDRV QZ\_NOSW\_NO\_INST\_ATTACH QZ\_NOSW\_LOW\_MEM

This function shall not be called in an interrupt context. None None Yes No Yes

##### Parameters

in	sess	Session handle (pointer to opaque instance and session data)
in	status	Pointer to QATZIP status structure

##### Return values

QZ_OK	Function executed successfully. The hardware based compression session has been created
QZ_PARAMS	*status is NULL

##### Precondition

None

##### Postcondition

None

##### Note

Only a synchronous version of this function is provided.

##### See Also

None

#### 4.1.5.11 QATZIP\_API int qzInit ( QzSession\_T \* sess, unsigned char sw\_backup )

Initialize QAT hardware

This function initializes the QAT hardware. This function is optional in the function calling sequence. If desired, this call can be made to avoid latency impact during the first call to qzDecompress or qzCompress, or to set the sw\_backup parameter explicitly. The input parameter sw\_backup specifies the behavior of the function and that of the functions called with the same session in the event there are insufficient resources to establish a QAT based compression or decompression session.

The required resources include access to the QAT hardware, contiguous pinned memory for mapping the hardware rings, and contiguous pinned memory for buffers.

This function shall not be called in an interrupt context. None This function will: 1) start the user space driver if necessary 2) allocate all hardware instances available Yes No Yes

## Parameters

in	<i>sess</i>	Session handle (pointer to opaque instance and session data.)
in	<i>sw_backup</i>	0 for no sw backup, 1 for sw backup

## Return values

<i>QZ_OK</i>	Function executed successfully. A hardware or software instance has been allocated to the calling process/thread
<i>QZ_DUPLICATE</i>	This process/thread already has a hardware instance
<i>QZ_PARAMS</i>	*sess is NULL
<i>QZ_NOSW_NO_HW</i>	No hardware and no software session being established
<i>QZ_NOSW_NO_MDRV</i>	No memory driver. No software session established
<i>QZ_NOSW_NO_INST_AT-TACH</i>	No instance available No software session established
<i>QZ_NOSW_LOW_MEM</i>	Not enough pinned memory available No software session established

## Precondition

None

## Postcondition

None

## Note

Only a synchronous version of this function is provided.

## See Also

None

## 4.1.5.12 QATZIP\_API void\* qzMalloc ( size\_t sz, int numa, int force\_pinned )

Allocate different types of memory

Allocate different types of memory.

This function shall not be called in an interrupt context. None None Yes No Yes

## Parameters

in	<i>sz</i>	Memory size to be allocated
in	<i>numa</i>	NUMA node from which to allocate memory
in	<i>force_pinned</i>	PINNED_MEM allocate contiguous memory COMMON_MEM allocate non-contiguous memory

## Return values

<i>NULL</i>	Fail to allocate memory
<i>address</i>	The address of allocated memory

## Precondition

None

**Postcondition**

None

**Note**

Only a synchronous version of this function is provided.

**See Also**

None

**4.1.5.13 QATZIP\_API int qzMemFindAddr ( unsigned char \* a )**

Check whether the address is available

Check whether the address is available.

This function shall not be called in an interrupt context. None None Yes No Yes

**Parameters**

in	a	Address need to be checked
----	---	----------------------------

**Return values**

1	The address is available
0	The address is not available

**Precondition**

None

**Postcondition**

None

**Note**

Only a synchronous version of this function is provided.

**See Also**

None

**4.1.5.14 QATZIP\_API int qzSetDefaults ( QzSessionParams\_T \* defaults )**

Set default QzSessionParams\_T value

Set default QzSessionParams\_T value.

This function shall not be called in an interrupt context. None None Yes No Yes



**Parameters**

<i>in</i>	<i>defaults</i>	The pointer to value to be set as default
-----------	-----------------	---

**Return values**

<i>QZ_OK</i>	Success on setting default value
<i>QZ_PARAM</i>	Fail to set default value

**Precondition**

None

**Postcondition**

None

**Note**

Only a synchronous version of this function is provided.

**See Also**

None

**4.1.5.15 QATZIP\_API int qzSetupSession ( QzSession\_T \* sess, QzSessionParams\_T \* params )**

Initialize a QATzip session

This function establishes a QAT session. This involves associating a hardware instance to the session, allocating buffers. If all of these activities can not be completed successfully, then this function will set up a software based session of param->sw\_backup that is set to 1.

Before this function is called, the hardware must have been successfully started via qzInit.

If \*sess includes an existing hardware or software session, then this session will be torn down before a new one is attempted.

This function shall not be called in an interrupt context. None None Yes No Yes

**Parameters**

<i>in</i>	<i>sess</i>	Session handle (pointer to opaque instance and session data)
<i>in</i>	<i>params</i>	Parameters for session

**Return values**

<i>QZ_OK</i>	Function executed successfully. A hardware or software based compression session has been created
<i>QZ_PARAMS</i>	*sess is NULL or member of params is invalid
<i>QZ_NOSW_NO_HW</i>	No hardware and no sw session being established
<i>QZ_NOSW_NO_MDRV</i>	No memory driver. No software session established
<i>QZ_NOSW_NO_INST_AT-TACH</i>	No instance available No software session established

<i>QZ_NO_LOW_MEM</i>	Not enough pinned memory available No software session established
----------------------	--

**Precondition**

None

**Postcondition**

None

**Note**

Only a synchronous version of this function is provided.

**See Also**

None

**4.1.5.16 QATZIP\_API int qzTeardownSession ( QzSession\_T \* sess )**

Deinitialize a QATZip session

This function disconnects a session from a hardware instance and deallocates buffers. If no session has been initialized, then no action will take place.

This function shall not be called in an interrupt context. None None Yes No Yes

**Parameters**

<i>in</i>	<i>sess</i>	Session handle (pointer to opaque instance and session data)
-----------	-------------	--

**Return values**

<i>QZ_OK</i>	Function executed successfully
<i>QZ_FAIL</i>	Function did not succeed
<i>QZ_PARAMS</i>	*sess is NULL or member of params is invalid

**Precondition**

None

**Postcondition**

None

**Note**

Only a synchronous version of this function is provided.

**See Also**

None

## Chapter 5

# Class Documentation

### 5.1 QzSession\_S Struct Reference

```
#include <qatzip.h>
```

#### Public Attributes

- signed long int [hw\\_session\\_stat](#)
- int [thd\\_sess\\_stat](#)
- void \* [internal](#)
- unsigned long [total\\_in](#)
- unsigned long [total\\_out](#)

#### 5.1.1 Detailed Description

QATZIP Session opaque data storage

This structure contains a pointer to a structure with session state.

#### 5.1.2 Member Data Documentation

##### 5.1.2.1 signed long int QzSession\_S::hw\_session\_stat

Filled in during initialization, session startup and decompression

##### 5.1.2.2 void\* QzSession\_S::internal

Session data is opaque to outside world

##### 5.1.2.3 int QzSession\_S::thd\_sess\_stat

Note process compression and decompression thread state

##### 5.1.2.4 unsigned long QzSession\_S::total\_in

Total processed input data length in this session

### 5.1.2.5 unsigned long QzSession\_S::total\_out

Total output data length in this session

The documentation for this struct was generated from the following file:

- [include/qatzip.h](#)

## 5.2 QzSessionParams\_S Struct Reference

```
#include <qatzip.h>
```

### Public Attributes

- [QzHuffmanHdr\\_T](#) huffman\_hdr
- [QzDirection\\_T](#) direction
- [QzDataFormat\\_T](#) data\_fmt
- unsigned int [comp\\_lvl](#)
- unsigned char [comp\\_algorithm](#)
- unsigned int [max\\_forks](#)
- unsigned char [sw\\_backup](#)
- unsigned int [hw\\_buff\\_sz](#)
- unsigned int [strm\\_buff\\_sz](#)
- unsigned int [input\\_sz\\_thrshold](#)
- unsigned int [req\\_cnt\\_thrshold](#)
- unsigned int [wait\\_cnt\\_thrshold](#)

### 5.2.1 Detailed Description

QATZIP Session Initialization parameters

This structure contains data for initializing a session.

### 5.2.2 Member Data Documentation

#### 5.2.2.1 unsigned char QzSessionParams\_S::comp\_algorithm

Compress/decompression algorithms

#### 5.2.2.2 unsigned int QzSessionParams\_S::comp\_lvl

Compression level 1 to 9

#### 5.2.2.3 QzDataFormat\_T QzSessionParams\_S::data\_fmt

Deflate, deflate with GZip or deflate with GZip ext

#### 5.2.2.4 QzDirection\_T QzSessionParams\_S::direction

Compress or decompress

## 5.2.2.5 QzHuffmanHdr\_T QzSessionParams\_S::huffman\_hdr

Dynamic or Static Huffman headers

## 5.2.2.6 unsigned int QzSessionParams\_S::hw\_buff\_sz

Default buffer size, must be a power of 2k 4K,8K,16K,32K,64K,128K

## 5.2.2.7 unsigned int QzSessionParams\_S::input\_sz\_threshold

Default threshold of compression service's input size for sw failover, if the size of input request less than the threshold, QATZip will route the request to software

## 5.2.2.8 unsigned int QzSessionParams\_S::max\_forks

Maximum forks permitted in the current thread 0 means no forking permitted

## 5.2.2.9 unsigned int QzSessionParams\_S::req\_cnt\_threshold

Set between 1 and NUM\_BUFF, default NUM\_BUFF NUM\_BUFF is defined in qatzip\_internal.h

## 5.2.2.10 unsigned int QzSessionParams\_S::strm\_buff\_sz

Stream buffer size between [1K .. 2M - 5K] Default strm\_buf\_sz equals to hw\_buff\_sz

## 5.2.2.11 unsigned char QzSessionParams\_S::sw\_backup

0 means no sw backup, 1 means sw backup

## 5.2.2.12 unsigned int QzSessionParams\_S::wait\_cnt\_threshold

When previous try failed, wait for specific number of call before retry to open device. Default threshold is 8

The documentation for this struct was generated from the following file:

- [include/qatzip.h](#)

## 5.3 QzStatus\_S Struct Reference

```
#include <qatzip.h>
```

### Public Attributes

- unsigned short int [qat\\_hw\\_count](#)
- unsigned char [qat\\_service\\_stated](#)
- unsigned char [qat\\_mem\\_drvr](#)
- unsigned char [qat\\_instance\\_attach](#)
- unsigned long int [memory\\_allocated](#)
- unsigned char [using\\_huge\\_pages](#)
- signed long int [hw\\_session\\_status](#)

- unsigned char [algo\\_sw](#) [[QZ\\_MAX\\_ALGORITHMS](#)]
- unsigned char [algo\\_hw](#) [[QZ\\_MAX\\_ALGORITHMS](#)]

### 5.3.1 Detailed Description

`QATZIP` status structure

This structure contains data relating to the status of QAT on the platform.

### 5.3.2 Member Data Documentation

#### 5.3.2.1 unsigned char `QzStatus_S::algo_hw`[[QZ\\_MAX\\_ALGORITHMS](#)]

Count of hardware devices supporting algorithms

#### 5.3.2.2 unsigned char `QzStatus_S::algo_sw`[[QZ\\_MAX\\_ALGORITHMS](#)]

Support software algorithms

#### 5.3.2.3 signed long int `QzStatus_S::hw_session_status`

One of QATZIP session status

#### 5.3.2.4 unsigned long int `QzStatus_S::memory_allocated`

Amount of memory allocated by this thread/process

#### 5.3.2.5 unsigned short int `QzStatus_S::qat_hw_count`

From PCI scan

#### 5.3.2.6 unsigned char `QzStatus_S::qat_instance_attach`

Is this thread/g\_process properly attached to an Instance?

#### 5.3.2.7 unsigned char `QzStatus_S::qat_mem_drvr`

1 if /dev/qat\_mem exists 2 if /dev/qat\_mem has been opened 0 otherwise

#### 5.3.2.8 unsigned char `QzStatus_S::qat_service_stated`

Check if the QAT service is running properly on at least one device

#### 5.3.2.9 unsigned char `QzStatus_S::using_huge_pages`

Are memory slabs coming from huge pages?

The documentation for this struct was generated from the following file:

- [include/qatzip.h](#)

## 5.4 QzStream\_S Struct Reference

```
#include <qatzip.h>
```

### Public Attributes

- unsigned int [in\\_sz](#)
- unsigned int [out\\_sz](#)
- unsigned char \* [in](#)
- unsigned char \* [out](#)
- unsigned int [pending\\_in](#)
- unsigned int [pending\\_out](#)
- [QzCrcType\\_T](#) [crc\\_type](#)
- unsigned int [crc\\_32](#)
- unsigned long long [reserved](#)
- void \* [opaque](#)

### 5.4.1 Detailed Description

QATZIP Stream data storage

This structure contains metadata needed for stream operation.

### 5.4.2 Member Data Documentation

#### 5.4.2.1 unsigned int QzStream\_S::crc\_32

Checksum value

#### 5.4.2.2 QzCrcType\_T QzStream\_S::crc\_type

Checksum type in Adler, CRC32 or none

#### 5.4.2.3 unsigned char\* QzStream\_S::in

Input data pointer set by application

#### 5.4.2.4 unsigned int QzStream\_S::in\_sz

Set by application, reset by QATZip to indicate consumed data

#### 5.4.2.5 void\* QzStream\_S::opaque

Internal storage managed by QATZip

#### 5.4.2.6 unsigned char\* QzStream\_S::out

Output data pointer set by application

#### 5.4.2.7 unsigned int QzStream\_S::out\_sz

Set by application, reset by QATZip to indicate processed data

#### 5.4.2.8 unsigned int QzStream\_S::pending\_in

Unprocessed bytes held in QATZip

#### 5.4.2.9 unsigned int QzStream\_S::pending\_out

Processed bytes held in QATZip

#### 5.4.2.10 unsigned long long QzStream\_S::reserved

CRC64 polynomial

The documentation for this struct was generated from the following file:

- [include/qatzip.h](#)



## Chapter 6

# File Documentation

### 6.1 include/qatzip.h File Reference

```
#include <string.h>
```

#### Classes

- struct [QzSessionParams\\_S](#)
- struct [QzSession\\_S](#)
- struct [QzStatus\\_S](#)
- struct [QzStream\\_S](#)

#### Macros

- #define [QATZIP\\_API](#)
- #define [QATZIP\\_API\\_VERSION\\_NUM\\_MAJOR](#) (1)
- #define [QATZIP\\_API\\_VERSION\\_NUM\\_MINOR](#) (2)
- #define [QATZIP\\_API\\_VERSION](#)
- #define [QZ\\_OK](#) (0)
- #define [QZ\\_DUPLICATE](#) (1)
- #define [QZ\\_FORCE\\_SW](#) (2)
- #define [QZ\\_PARAMS](#) (-1)
- #define [QZ\\_FAIL](#) (-2)
- #define [QZ\\_BUF\\_ERROR](#) (-3)
- #define [QZ\\_DATA\\_ERROR](#) (-4)
- #define [QZ\\_NO\\_HW](#) (11)
- #define [QZ\\_NO\\_MDRV](#) (12)
- #define [QZ\\_NO\\_INST\\_ATTACH](#) (13)
- #define [QZ\\_LOW\\_MEM](#) (14)
- #define [QZ\\_LOW\\_DEST\\_MEM](#) (15)
- #define [QZ\\_NONE](#) (100)
- #define [QZ\\_NOSW\\_NO\\_HW](#) (-101)
- #define [QZ\\_NOSW\\_NO\\_MDRV](#) (-102)
- #define [QZ\\_NOSW\\_NO\\_INST\\_ATTACH](#) (-103)
- #define [QZ\\_NOSW\\_LOW\\_MEM](#) (-104)
- #define [QZ\\_MAX\\_ALGORITHMS](#) ((int)255)
- #define [QZ\\_DEFLATE](#) ((unsigned char)8)
- #define [MIN](#)(a, b) (((a)<(b))?(a):(b))

- `#define QZ_MEMCPY(dest, src, dest_sz, src_sz) memcpy((void *) (dest), (void *) (src), (size_t)MIN(dest_sz, src_sz))`
- `#define QZ_HUFF_HDR_DEFAULT QZ_DYNAMIC_HDR`
- `#define QZ_DIRECTION_DEFAULT QZ_DIR_BOTH`
- `#define QZ_DATA_FORMAT_DEFAULT QZ_DEFLATE_GZIP_EXT`
- `#define QZ_COMP_LEVEL_DEFAULT 1`
- `#define QZ_COMP_ALGOL_DEFAULT QZ_DEFLATE`
- `#define QZ_POLL_SLEEP_DEFAULT 10`
- `#define QZ_MAX_FORK_DEFAULT 3`
- `#define QZ_SW_BACKUP_DEFAULT 1`
- `#define QZ_HW_BUFF_SZ (64*1024)`
- `#define QZ_HW_BUFF_MIN_SZ (1*1024)`
- `#define QZ_HW_BUFF_MAX_SZ (512*1024)`
- `#define QZ_STRM_BUFF_SZ_DEFAULT QZ_HW_BUFF_SZ`
- `#define QZ_STRM_BUFF_MIN_SZ (1*1024)`
- `#define QZ_STRM_BUFF_MAX_SZ (2*1024*1024 - 5*1024)`
- `#define QZ_COMP_THRESHOLD_DEFAULT 1024`
- `#define QZ_COMP_THRESHOLD_MINIMUM 128`
- `#define QZ_REQ_THRESHOLD_MINIMUM 1`
- `#define QZ_REQ_THRESHOLD_MAXIMUM NUM_BUFF`
- `#define QZ_REQ_THRESHOLD_DEFAULT QZ_REQ_THRESHOLD_MAXIMUM`
- `#define QZ_WAIT_CNT_THRESHOLD_DEFAULT 8`
- `#define QZ_DEFLATE_COMP_LVL_MINIMUM (1)`
- `#define QZ_DEFLATE_COMP_LVL_MAXIMUM (9)`
- `#define QZ_SKID_PAD_SZ 48`
- `#define QZ_COMPRESSED_SZ_OF_EMPTY_FILE 34`

## Typedefs

- `typedef enum QzHuffmanHdr_E QzHuffmanHdr_T`
- `typedef enum PinMem_E PinMem_T`
- `typedef enum QzDirection_E QzDirection_T`
- `typedef enum QzDataFormat_E QzDataFormat_T`
- `typedef enum QzCrcType_E QzCrcType_T`
- `typedef struct QzSessionParams_S QzSessionParams_T`
- `typedef struct QzSession_S QzSession_T`
- `typedef struct QzStatus_S QzStatus_T`
- `typedef struct QzStream_S QzStream_T`

## Enumerations

- `enum QzHuffmanHdr_E { QZ_DYNAMIC_HDR = 0, QZ_STATIC_HDR }`
- `enum PinMem_E { COMMON_MEM = 0, PINNED_MEM }`
- `enum QzDirection_E { QZ_DIR_COMPRESS = 0, QZ_DIR_DECOMPRESS, QZ_DIR_BOTH }`
- `enum QzDataFormat_E { QZ_DEFLATE_4B = 0, QZ_DEFLATE_GZIP, QZ_DEFLATE_GZIP_EXT, QZ_DEFLATE_RAW, QZ_FMT_NUM }`
- `enum QzCrcType_E { QZ_CRC32 = 0, QZ_ADLER, NONE }`

## Functions

- QATZIP\_API int [qzInit](#) (QzSession\_T \*sess, unsigned char sw\_backup)
- QATZIP\_API int [qzSetupSession](#) (QzSession\_T \*sess, QzSessionParams\_T \*params)
- QATZIP\_API int [qzCompress](#) (QzSession\_T \*sess, const unsigned char \*src, unsigned int \*src\_len, unsigned char \*dest, unsigned int \*dest\_len, unsigned int last)
- QATZIP\_API int [qzCompressCrc](#) (QzSession\_T \*sess, const unsigned char \*src, unsigned int \*src\_len, unsigned char \*dest, unsigned int \*dest\_len, unsigned int last, unsigned long \*crc)
- QATZIP\_API int [qzDecompress](#) (QzSession\_T \*sess, const unsigned char \*src, unsigned int \*src\_len, unsigned char \*dest, unsigned int \*dest\_len)
- QATZIP\_API int [qzTeardownSession](#) (QzSession\_T \*sess)
- QATZIP\_API int [qzClose](#) (QzSession\_T \*sess)
- QATZIP\_API int [qzGetStatus](#) (QzSession\_T \*sess, QzStatus\_T \*status)
- QATZIP\_API unsigned int [qzMaxCompressedLength](#) (unsigned int src\_sz, QzSession\_T \*sess)
- QATZIP\_API int [qzSetDefaults](#) (QzSessionParams\_T \*defaults)
- QATZIP\_API int [qzGetDefaults](#) (QzSessionParams\_T \*defaults)
- QATZIP\_API void \* [qzMalloc](#) (size\_t sz, int numa, int force\_pinned)
- QATZIP\_API void [qzFree](#) (void \*m)
- QATZIP\_API int [qzMemFindAddr](#) (unsigned char \*a)
- QATZIP\_API int [qzCompressStream](#) (QzSession\_T \*sess, QzStream\_T \*strm, unsigned int last)
- QATZIP\_API int [qzDecompressStream](#) (QzSession\_T \*sess, QzStream\_T \*strm, unsigned int last)
- QATZIP\_API int [qzEndStream](#) (QzSession\_T \*sess, QzStream\_T \*strm)

### 6.1.1 Macro Definition Documentation

6.1.1.1 `#define MIN( a, b ) (((a)<(b))?(a):(b))`

6.1.1.2 `#define QATZIP_API`

These macros define how the project will be built QATZIP\_LINK\_DLL must be defined if linking the DLL QATZIP\_BUILD\_DLL must be defined when building a DLL No definition required if building the project as static library

6.1.1.3 `#define QATZIP_API_VERSION`

**Value:**

```
(QATZIP_API_VERSION_NUM_MAJOR * 10000 + \
    QATZIP_API_VERSION_NUM_MINOR
    * 100)
```

6.1.1.4 `#define QZ_BUF_ERROR (-3)`

Insufficient buffer error

6.1.1.5 `#define QZ_COMP_ALGOL_DEFAULT QZ_DEFLATE`

6.1.1.6 `#define QZ_COMP_LEVEL_DEFAULT 1`

6.1.1.7 `#define QZ_COMP_THRESHOLD_DEFAULT 1024`

6.1.1.8 `#define QZ_COMP_THRESHOLD_MINIMUM 128`

6.1.1.9 `#define QZ_COMPRESSED_SZ_OF_EMPTY_FILE 34`

6.1.1.10 `#define QZ_DATA_ERROR (-4)`

Input data was corrupted

6.1.1.11 `#define QZ_DATA_FORMAT_DEFAULT QZ_DEFLATE_GZIP_EXT`

6.1.1.12 `#define QZ_DEFLATE ((unsigned char)8)`

6.1.1.13 `#define QZ_DEFLATE_COMP_LVL_MAXIMUM (9)`

6.1.1.14 `#define QZ_DEFLATE_COMP_LVL_MINIMUM (1)`

6.1.1.15 `#define QZ_DIRECTION_DEFAULT QZ_DIR_BOTH`

6.1.1.16 `#define QZ_DUPLICATE (1)`

Can not process function again. No failure

6.1.1.17 `#define QZ_FAIL (-2)`

Unspecified error

6.1.1.18 `#define QZ_FORCE_SW (2)`

Using SW: Switch to software because of previous block

6.1.1.19 `#define QZ_HUFF_HDR_DEFAULT QZ_DYNAMIC_HDR`

6.1.1.20 `#define QZ_HW_BUFF_MAX_SZ (512*1024)`

6.1.1.21 `#define QZ_HW_BUFF_MIN_SZ (1*1024)`

6.1.1.22 `#define QZ_HW_BUFF_SZ (64*1024)`

6.1.1.23 `#define QZ_LOW_DEST_MEM (15)`

Using SW: Not enough pinned memory for dest buffer

6.1.1.24 `#define QZ_LOW_MEM (14)`

Using SW: Not enough pinned memory

6.1.1.25 `#define QZ_MAX_ALGORITHMS ((int)255)`

6.1.1.26 `#define QZ_MAX_FORK_DEFAULT 3`

6.1.1.27 `#define QZ_MEMCPY( dest, src, dest_sz, src_sz ) memcpy((void*)(dest), (void*)(src), (size_t)MIN(dest_sz, src_sz))`

**6.1.1.28 #define QZ\_NO\_HW (11)**

Using SW: No QAT HW detected

**6.1.1.29 #define QZ\_NO\_INST\_ATTACH (13)**

Using SW: Could not attach to an instance

**6.1.1.30 #define QZ\_NO\_MDRV (12)**

Using SW: No memory driver detected

**6.1.1.31 #define QZ\_NONE (100)**

Device uninitialized

**6.1.1.32 #define QZ\_NOSW\_LOW\_MEM (-104)**

Not using SW: not enough pinned memory

**6.1.1.33 #define QZ\_NOSW\_NO\_HW (-101)**

Not using SW: No QAT HW detected

**6.1.1.34 #define QZ\_NOSW\_NO\_INST\_ATTACH (-103)**

Not using SW: Could not attach to instance

**6.1.1.35 #define QZ\_NOSW\_NO\_MDRV (-102)**

Not using SW: No memory driver detected

**6.1.1.36 #define QZ\_PARAMS (-1)**

Invalid parameter in function call

**6.1.1.37 #define QZ\_POLL\_SLEEP\_DEFAULT 10****6.1.1.38 #define QZ\_REQ\_THRESHOLD\_DEFAULT QZ\_REQ\_THRESHOLD\_MAXIMUM****6.1.1.39 #define QZ\_REQ\_THRESHOLD\_MAXIMUM NUM\_BUFF****6.1.1.40 #define QZ\_REQ\_THRESHOLD\_MINIMUM 1****6.1.1.41 #define QZ\_STRM\_BUFF\_MAX\_SZ (2\*1024\*1024 - 5\*1024)****6.1.1.42 #define QZ\_STRM\_BUFF\_MIN\_SZ (1\*1024)****6.1.1.43 #define QZ\_STRM\_BUFF\_SZ\_DEFAULT QZ\_HW\_BUFF\_SZ**

6.1.1.44 `#define QZ_SW_BACKUP_DEFAULT 1`

6.1.1.45 `#define QZ_WAIT_CNT_THRESHOLD_DEFAULT 8`

## 6.1.2 Function Documentation

6.1.2.1 `QATZIP_API unsigned int qzMaxCompressedLength ( unsigned int src_sz, QzSession_T * sess )`

# Index

- algo\_hw
  - QzStatus\_S, 30
- algo\_sw
  - QzStatus\_S, 30
- COMMON\_MEM
  - Data Compression API, 11
- comp\_algorithm
  - QzSessionParams\_S, 28
- comp\_lvl
  - QzSessionParams\_S, 28
- crc\_32
  - QzStream\_S, 31
- crc\_type
  - QzStream\_S, 31
- Data Compression API
  - COMMON\_MEM, 11
  - NONE, 11
  - PINNED\_MEM, 11
  - QZ\_ADLER, 11
  - QZ\_CRC32, 11
  - QZ\_DEFLATE\_4B, 11
  - QZ\_DEFLATE\_GZIP, 11
  - QZ\_DEFLATE\_GZIP\_EXT, 11
  - QZ\_DEFLATE\_RAW, 11
  - QZ\_DIR\_BOTH, 12
  - QZ\_DIR\_COMPRESS, 12
  - QZ\_DIR\_DECOMPRESS, 12
  - QZ\_DYNAMIC\_HDR, 12
  - QZ\_FMT\_NUM, 11
  - QZ\_STATIC\_HDR, 12
- Data Compression API, 7
  - PinMem\_E, 11
  - PinMem\_T, 9
  - QZ\_OK, 8
  - QZ\_SKID\_PAD\_SZ, 8
  - qzClose, 13
  - qzCompress, 13
  - qzCompressCrc, 14
  - qzCompressStream, 15
  - QzCrcType\_E, 11
  - QzCrcType\_T, 9
  - QzDataFormat\_E, 11
  - QzDataFormat\_T, 9
  - qzDecompress, 17
  - qzDecompressStream, 18
  - QzDirection\_E, 11
  - QzDirection\_T, 9
  - qzEndStream, 19
  - qzFree, 19
  - qzGetDefaults, 21
  - qzGetStatus, 21
  - QzHuffmanHdr\_E, 12
  - QzHuffmanHdr\_T, 10
  - qzInit, 22
  - qzMalloc, 23
  - qzMemFindAddr, 24
  - QzSession\_T, 10
  - QzSessionParams\_T, 10
  - qzSetDefaults, 24
  - qzSetupSession, 25
  - QzStatus\_T, 10
  - QzStream\_T, 11
  - qzTeardownSession, 26
- data\_fmt
  - QzSessionParams\_S, 28
- direction
  - QzSessionParams\_S, 28
- huffman\_hdr
  - QzSessionParams\_S, 28
- hw\_buff\_sz
  - QzSessionParams\_S, 29
- hw\_session\_stat
  - QzSession\_S, 27
- hw\_session\_status
  - QzStatus\_S, 30
- in
  - QzStream\_S, 31
- in\_sz
  - QzStream\_S, 31
- include/qatzip.h, 33
- input\_sz\_threshold
  - QzSessionParams\_S, 29
- internal
  - QzSession\_S, 27
- MIN
  - qatzip.h, 35
- max\_forks
  - QzSessionParams\_S, 29
- memory\_allocated
  - QzStatus\_S, 30
- NONE
  - Data Compression API, 11
- opaque
  - QzStream\_S, 31

- out
  - QzStream\_S, [31](#)
- out\_sz
  - QzStream\_S, [31](#)
- PINNED\_MEM
  - Data Compression API, [11](#)
- pending\_in
  - QzStream\_S, [32](#)
- pending\_out
  - QzStream\_S, [32](#)
- PinMem\_E
  - Data Compression API, [11](#)
- PinMem\_T
  - Data Compression API, [9](#)
- QZ\_ADLER
  - Data Compression API, [11](#)
- QZ\_CRC32
  - Data Compression API, [11](#)
- QZ\_DEFLATE\_4B
  - Data Compression API, [11](#)
- QZ\_DEFLATE\_GZIP
  - Data Compression API, [11](#)
- QZ\_DEFLATE\_GZIP\_EXT
  - Data Compression API, [11](#)
- QZ\_DEFLATE\_RAW
  - Data Compression API, [11](#)
- QZ\_DIR\_BOTH
  - Data Compression API, [12](#)
- QZ\_DIR\_COMPRESS
  - Data Compression API, [12](#)
- QZ\_DIR\_DECOMPRESS
  - Data Compression API, [12](#)
- QZ\_DYNAMIC\_HDR
  - Data Compression API, [12](#)
- QZ\_FMT\_NUM
  - Data Compression API, [11](#)
- QZ\_STATIC\_HDR
  - Data Compression API, [12](#)
- QATZIP\_API
  - qatzip.h, [35](#)
- QATZIP\_API\_VERSION
  - qatzip.h, [35](#)
- QZ\_BUF\_ERROR
  - qatzip.h, [35](#)
- QZ\_DATA\_ERROR
  - qatzip.h, [36](#)
- QZ\_DEFLATE
  - qatzip.h, [36](#)
- QZ\_DIRECTION\_DEFAULT
  - qatzip.h, [36](#)
- QZ\_DUPLICATE
  - qatzip.h, [36](#)
- QZ\_FAIL
  - qatzip.h, [36](#)
- QZ\_FORCE\_SW
  - qatzip.h, [36](#)
- QZ\_HUFF\_HDR\_DEFAULT
  - qatzip.h, [36](#)
- QZ\_HW\_BUFF\_MAX\_SZ
  - qatzip.h, [36](#)
- QZ\_HW\_BUFF\_MIN\_SZ
  - qatzip.h, [36](#)
- QZ\_HW\_BUFF\_SZ
  - qatzip.h, [36](#)
- QZ\_LOW\_DEST\_MEM
  - qatzip.h, [36](#)
- QZ\_LOW\_MEM
  - qatzip.h, [36](#)
- QZ\_MAX\_ALGORITHMS
  - qatzip.h, [36](#)
- QZ\_MAX\_FORK\_DEFAULT
  - qatzip.h, [36](#)
- QZ\_MEMCPY
  - qatzip.h, [36](#)
- QZ\_NO\_HW
  - qatzip.h, [36](#)
- QZ\_NO\_INST\_ATTACH
  - qatzip.h, [37](#)
- QZ\_NO\_MDRV
  - qatzip.h, [37](#)
- QZ\_NONE
  - qatzip.h, [37](#)
- QZ\_NOSW\_LOW\_MEM
  - qatzip.h, [37](#)
- QZ\_NOSW\_NO\_HW
  - qatzip.h, [37](#)
- QZ\_NOSW\_NO\_MDRV
  - qatzip.h, [37](#)
- QZ\_OK
  - Data Compression API, [8](#)
- QZ\_PARAMS
  - qatzip.h, [37](#)
- QZ\_SKID\_PAD\_SZ
  - Data Compression API, [8](#)
- qat\_hw\_count
  - QzStatus\_S, [30](#)
- qat\_instance\_attach
  - QzStatus\_S, [30](#)
- qat\_mem\_drvr
  - QzStatus\_S, [30](#)
- qat\_service\_stated
  - QzStatus\_S, [30](#)
- qatzip.h
  - MIN, [35](#)
  - QATZIP\_API, [35](#)
  - QATZIP\_API\_VERSION, [35](#)
  - QZ\_BUF\_ERROR, [35](#)
  - QZ\_DATA\_ERROR, [36](#)
  - QZ\_DEFLATE, [36](#)
  - QZ\_DUPLICATE, [36](#)
  - QZ\_FAIL, [36](#)
  - QZ\_FORCE\_SW, [36](#)
  - QZ\_HW\_BUFF\_MAX\_SZ, [36](#)
  - QZ\_HW\_BUFF\_MIN\_SZ, [36](#)
  - QZ\_HW\_BUFF\_SZ, [36](#)



- QZ\_LOW\_DEST\_MEM, [36](#)
- QZ\_LOW\_MEM, [36](#)
- QZ\_MAX\_ALGORITHMS, [36](#)
- QZ\_MEMCPY, [36](#)
- QZ\_NO\_HW, [36](#)
- QZ\_NO\_INST\_ATTACH, [37](#)
- QZ\_NO\_MDRV, [37](#)
- QZ\_NONE, [37](#)
- QZ\_NOSW\_LOW\_MEM, [37](#)
- QZ\_NOSW\_NO\_HW, [37](#)
- QZ\_NOSW\_NO\_MDRV, [37](#)
- QZ\_PARAMS, [37](#)
- qzMaxCompressedLength, [38](#)
- qzClose
  - Data Compression API, [13](#)
- qzCompress
  - Data Compression API, [13](#)
- qzCompressCrc
  - Data Compression API, [14](#)
- qzCompressStream
  - Data Compression API, [15](#)
- QzCrcType\_E
  - Data Compression API, [11](#)
- QzCrcType\_T
  - Data Compression API, [9](#)
- QzDataFormat\_E
  - Data Compression API, [11](#)
- QzDataFormat\_T
  - Data Compression API, [9](#)
- qzDecompress
  - Data Compression API, [17](#)
- qzDecompressStream
  - Data Compression API, [18](#)
- QzDirection\_E
  - Data Compression API, [11](#)
- QzDirection\_T
  - Data Compression API, [9](#)
- qzEndStream
  - Data Compression API, [19](#)
- qzFree
  - Data Compression API, [19](#)
- qzGetDefaults
  - Data Compression API, [21](#)
- qzGetStatus
  - Data Compression API, [21](#)
- QzHuffmanHdr\_E
  - Data Compression API, [12](#)
- QzHuffmanHdr\_T
  - Data Compression API, [10](#)
- qzInit
  - Data Compression API, [22](#)
- qzMalloc
  - Data Compression API, [23](#)
- qzMaxCompressedLength
  - qatzip.h, [38](#)
- qzMemFindAddr
  - Data Compression API, [24](#)
- QzSession\_S, [27](#)
- hw\_session\_stat, [27](#)
- internal, [27](#)
- thd\_sess\_stat, [27](#)
- total\_in, [27](#)
- total\_out, [27](#)
- QzSession\_T
  - Data Compression API, [10](#)
- QzSessionParams\_S, [28](#)
  - comp\_algorithm, [28](#)
  - comp\_lvl, [28](#)
  - data\_fmt, [28](#)
  - direction, [28](#)
  - huffman\_hdr, [28](#)
  - hw\_buff\_sz, [29](#)
  - input\_sz\_thrshold, [29](#)
  - max\_forks, [29](#)
  - req\_cnt\_thrshold, [29](#)
  - strm\_buff\_sz, [29](#)
  - sw\_backup, [29](#)
  - wait\_cnt\_thrshold, [29](#)
- QzSessionParams\_T
  - Data Compression API, [10](#)
- qzSetDefaults
  - Data Compression API, [24](#)
- qzSetupSession
  - Data Compression API, [25](#)
- QzStatus\_S, [29](#)
  - algo\_hw, [30](#)
  - algo\_sw, [30](#)
  - hw\_session\_status, [30](#)
  - memory\_allocated, [30](#)
  - qat\_hw\_count, [30](#)
  - qat\_instance\_attach, [30](#)
  - qat\_mem\_drvr, [30](#)
  - qat\_service\_stated, [30](#)
  - using\_huge\_pages, [30](#)
- QzStatus\_T
  - Data Compression API, [10](#)
- QzStream\_S, [31](#)
  - crc\_32, [31](#)
  - crc\_type, [31](#)
  - in, [31](#)
  - in\_sz, [31](#)
  - opaque, [31](#)
  - out, [31](#)
  - out\_sz, [31](#)
  - pending\_in, [32](#)
  - pending\_out, [32](#)
  - reserved, [32](#)
- QzStream\_T
  - Data Compression API, [11](#)
- qzTeardownSession
  - Data Compression API, [26](#)
- req\_cnt\_thrshold
  - QzSessionParams\_S, [29](#)
- reserved
  - QzStream\_S, [32](#)

strm\_buff\_sz  
    QzSessionParams\_S, [29](#)  
sw\_backup  
    QzSessionParams\_S, [29](#)  
  
thd\_sess\_stat  
    QzSession\_S, [27](#)  
total\_in  
    QzSession\_S, [27](#)  
total\_out  
    QzSession\_S, [27](#)  
  
using\_huge\_pages  
    QzStatus\_S, [30](#)  
  
wait\_cnt\_thrshold  
    QzSessionParams\_S, [29](#)