

QATzip

Generated by Doxygen 1.8.5

Tue Nov 28 2017 11:26:47

Contents

1	Module Index	1
1.1	Modules	1
2	Class Index	3
2.1	Class List	3
3	File Index	5
3.1	File List	5
4	Module Documentation	7
4.1	Data Compression API	7
4.1.1	Detailed Description	8
4.1.2	Macro Definition Documentation	8
4.1.2.1	QZ_OK	8
4.1.2.2	QZ_SKID_PAD_SZ	8
4.1.3	Typedef Documentation	9
4.1.3.1	QzDirection_T	9
4.1.3.2	QzHuffmanHdr_T	9
4.1.3.3	QzSession_T	9
4.1.3.4	QzSessionParams_T	9
4.1.3.5	QzStatus_T	10
4.1.4	Enumeration Type Documentation	10
4.1.4.1	QzDirection_E	10
4.1.4.2	QzHuffmanHdr_E	10
4.1.5	Function Documentation	11
4.1.5.1	qzClose	11
4.1.5.2	qzCompress	11
4.1.5.3	qzCompressCrc	12
4.1.5.4	qzDecompress	13
4.1.5.5	qzFree	14
4.1.5.6	qzGetDefaults	14
4.1.5.7	qzGetStatus	15

4.1.5.8	qzInit	16
4.1.5.9	qzMalloc	17
4.1.5.10	qzMemFindAddr	18
4.1.5.11	qzSetDefaults	19
4.1.5.12	qzSetupSession	19
4.1.5.13	qzTeardownSession	20
5	Class Documentation	21
5.1	QzSession_S Struct Reference	21
5.1.1	Detailed Description	21
5.1.2	Member Data Documentation	21
5.1.2.1	hw_session_stat	21
5.1.2.2	internal	21
5.1.2.3	thd_sess_stat	21
5.1.2.4	total_in	21
5.1.2.5	total_out	22
5.2	QzSessionParams_S Struct Reference	22
5.2.1	Detailed Description	22
5.2.2	Member Data Documentation	22
5.2.2.1	comp_algorithm	22
5.2.2.2	comp_lvl	22
5.2.2.3	direction	22
5.2.2.4	huffman_hdr	22
5.2.2.5	hw_buff_sz	22
5.2.2.6	input_sz_thrshold	23
5.2.2.7	max_forks	23
5.2.2.8	poll_sleep	23
5.2.2.9	req_cnt_thrshold	23
5.2.2.10	sw_backup	23
5.3	QzStatus_S Struct Reference	23
5.3.1	Detailed Description	23
5.3.2	Member Data Documentation	23
5.3.2.1	algo_hw	23
5.3.2.2	algo_sw	24
5.3.2.3	hw_session_status	24
5.3.2.4	memory_allocated	24
5.3.2.5	qat_hw_count	24
5.3.2.6	qat_instance_attach	24
5.3.2.7	qat_mem_drvr	24
5.3.2.8	qat_service_stated	24

5.3.2.9	using_huge_pages	24
6	File Documentation	25
6.1	include/qatzip.h File Reference	25
6.1.1	Macro Definition Documentation	26
6.1.1.1	MIN	26
6.1.1.2	QZ_BUF_ERROR	26
6.1.1.3	QZ_COMP_ALGOL_DEFAULT	27
6.1.1.4	QZ_COMP_LEVEL_DEFAULT	27
6.1.1.5	QZ_COMP_THRESHOLD_DEFAULT	27
6.1.1.6	QZ_COMP_THRESHOLD_MINIMUM	27
6.1.1.7	QZ_DATA_ERROR	27
6.1.1.8	QZ_DEFLATE	27
6.1.1.9	QZ_DIRECTION_DEFAULT	27
6.1.1.10	QZ_DUPLICATE	27
6.1.1.11	QZ_FAIL	27
6.1.1.12	QZ_FORCE_SW	27
6.1.1.13	QZ_HUFF_HDR_DEFAULT	27
6.1.1.14	QZ_HW_BUFF_MAX_SZ	27
6.1.1.15	QZ_HW_BUFF_MIN_SZ	27
6.1.1.16	QZ_HW_BUFF_SZ	27
6.1.1.17	QZ_LOW_MEM	27
6.1.1.18	QZ_LZ4	27
6.1.1.19	QZ_MAX_ALGORITHMS	27
6.1.1.20	QZ_MAX_FORK_DEFAULT	27
6.1.1.21	QZ_MEMCPY	27
6.1.1.22	QZ_NO_HW	27
6.1.1.23	QZ_NO_INST_ATTACH	28
6.1.1.24	QZ_NO_MDRV	28
6.1.1.25	QZ_NOSW_LOW_MEM	28
6.1.1.26	QZ_NOSW_NO_HW	28
6.1.1.27	QZ_NOSW_NO_INST_ATTACH	28
6.1.1.28	QZ_NOSW_NO_MDRV	28
6.1.1.29	QZ_PARAMS	28
6.1.1.30	QZ_POLL_SLEEP_DEFAULT	28
6.1.1.31	QZ_REQ_THRESHOLD_DEFAULT	28
6.1.1.32	QZ_REQ_THRESHOLD_MAXINUM	28
6.1.1.33	QZ_REQ_THRESHOLD_MINIMUM	28
6.1.1.34	QZ_SNAPPY	28
6.1.1.35	QZ_SW_BACKUP_DEFAULT	28

6.1.2	Function Documentation	28
6.1.2.1	qzMaxCompressedLength	28
Index		29

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:

QzSession_S	21
QzSessionParams_S	22
QzStatus_S	23

Chapter 3

File Index

3.1 File List

Here is a list of all files with brief descriptions:

include/ qatzip.h	25
---	----

Chapter 4

Module Documentation

4.1 Data Compression API

Classes

- struct [QzSessionParams_S](#)
- struct [QzSession_S](#)
- struct [QzStatus_S](#)

Macros

- `#define QZ_OK (0)`
- `#define QZ_SKID_PAD_SZ 48`

Typedefs

- typedef enum [QzHuffmanHdr_E](#) [QzHuffmanHdr_T](#)
- typedef enum [QzDirection_E](#) [QzDirection_T](#)
- typedef struct [QzSessionParams_S](#) [QzSessionParams_T](#)
- typedef struct [QzSession_S](#) [QzSession_T](#)
- typedef struct [QzStatus_S](#) [QzStatus_T](#)

Enumerations

- enum [QzHuffmanHdr_E](#) { [QZ_DYNAMIC_HDR](#) = 0, [QZ_STATIC_HDR](#) }
- enum [QzDirection_E](#) { [QZ_DIR_COMPRESS](#) = 0, [QZ_DIR_DECOMPRESS](#), [QZ_DIR_BOTH](#) }

Functions

- int [qzInit](#) ([QzSession_T](#) *sess, unsigned char sw_backup)
- int [qzSetupSession](#) ([QzSession_T](#) *sess, [QzSessionParams_T](#) *params)
- int [qzCompress](#) ([QzSession_T](#) *sess, const unsigned char *src, unsigned int *src_len, unsigned char *dest, unsigned int *dest_len, unsigned int last)
- 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)
- int [qzDecompress](#) ([QzSession_T](#) *sess, const unsigned char *src, unsigned int *src_len, unsigned char *dest, unsigned int *dest_len)
- int [qzTeardownSession](#) ([QzSession_T](#) *sess)

- int [qzClose](#) ([QzSession_T](#) *sess)
- int [qzGetStatus](#) ([QzSession_T](#) *sess, [QzStatus_T](#) *status)
- int [qzSetDefaults](#) ([QzSessionParams_T](#) *defaults)
- int [qzGetDefaults](#) ([QzSessionParams_T](#) *defaults)
- void * [qzMalloc](#) (size_t sz, int numa, int force_pinned)
- void [qzFree](#) (void *m)
- int [qzMemFindAddr](#) (unsigned char *a)

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 QZ_OK (0)

QATZIP Session Status definitions and function return codes

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

4.1.2.2 #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

<i>in</i>	<i>src_sz</i>	Input data length in byte.
-----------	---------------	----------------------------

Return values

<i>dest_sz</i>	Max compressed data output length in byte. When <i>src_sz</i> equal to 0, 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 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.

4.1.3.2 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 not 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_c, sw_backup); qzSetupSession(&sess_c, &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 without 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 cleanup until the application exits.

Supported Huffman Headers

This enumerated list identifies the Huffman header types supported by QATZip

4.1.3.3 typedef struct QzSession_S QzSession_T

QATZIP Session opaque data storage

This structure contains a pointer to a structure with session state

4.1.3.4 typedef struct QzSessionParams_S QzSessionParams_T

QATZIP Session Initialization parameters

This structure contains data for initializing a session

4.1.3.5 typedef struct QzStatus_S QzStatus_T

QATZIP status structure

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

4.1.4 Enumeration Type Documentation

4.1.4.1 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

4.1.4.2 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 not 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_c, sw_backup); qzSetupSession(&sess_c, &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 without 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 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	pointer to 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 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 hw based session or a software based session is available. If no session has been established - as indicated by the contents of *sess - then this app will attempt to set up a session using qzinit and qzSetupSession.

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
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 `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 check sum

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

This function will place completed compression blocks in the output buffer and put CRC32 checksum for compressed input data in user provided bufer *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
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 `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 hw based session or a software based session is available. If no session has been established - as indicated by the contents of *sess - then this app 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 RFC1952.

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

Parameters

in	<i>sess</i>	Session handle
in	<i>src</i>	point to source buffer
in	<i>src_len</i>	length of source buffer. Modified to length of processed compressed data when function returns
in	<i>dest</i>	point to destination buffer
in,out	<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

4.1.5.5 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

4.1.5.6 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>	Success on getting default value.
<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

4.1.5.7 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 hw instance, 0 otherwise memory_alloced 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

<i>in</i>	<i>dcInstance</i>	Instance handle derived from discovery functions
-----------	-------------------	--

Return values

<i>QZ_OK</i>	Function executed successfully. A hw based compression session has been created.
<i>QZ_PARAMS</i>	*status is NULL

Precondition

None

Postcondition

None

Note

Only a synchronous version of this function is provided.

See Also

None

4.1.5.8 int qzlnit (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.

Required resources include access to the QAT hardware, contiguous pinned memory for mmaping 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 hw instances available Yes No Yes

Parameters

in	sess	pointer to opaque instance and session data.
in	sw_backup	0 for no sw backup, 1 for sw backup

Return values

QZ_OK	Function executed successfully. A hw or sw instance has been allocated to the calling process/thread.
QZ_DUPLICATE	This process/thread already has a hw instance
QZ_PARAMS	*sess is NULL
QZ_NOSW_NO_HW	No hardware and no sw session being established
QZ_NOSW_NO_MDRV	No memory driver. No software session established
QZ_NOSW_NO_INST_AT-TACH	No instance avail. No software session established
QZ_NOSW_LOW_MEM	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

4.1.5.9 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

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

Return values

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

Precondition

None

Postcondition

None

Note

Only a synchronous version of this function is provided.

See Also

None

4.1.5.10 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

<i>in</i>	<i>a</i>	Address need to be checked
-----------	----------	----------------------------

Return values

<i>1</i>	The Address is available
<i>0</i>	The address is not available

Precondition

None

Postcondition

None

Note

Only a synchronous version of this function is provided.

See Also

None

4.1.5.11 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

in	defaults	The pointer to value to be set as default.
----	----------	--

Return values

QZ_OK	Success on setting default value.
QZ_PARAM	Fail to set default value.

Precondition

None

Postcondition

None

Note

Only a synchronous version of this function is provided.

See Also

None

4.1.5.12 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 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

in	sess	Session handle
in	params	Parameters for session

Return values

QZ_OK	Function executed successfully. A hw or sw 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 avail. 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

4.1.5.13 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

in	sess	Session handle
----	------	----------------

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](#)
- unsigned int [comp_lvl](#)
- unsigned char [comp_algorithm](#)
- unsigned int [poll_sleep](#)
- unsigned int [max_forks](#)
- unsigned char [sw_backup](#)
- unsigned int [hw_buff_sz](#)
- unsigned int [input_sz_thrshold](#)
- unsigned int [req_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

5.2.2.2 unsigned int QzSessionParams_S::comp_lvl

Compression level 1..9

5.2.2.3 QzDirection_T QzSessionParams_S::direction

compress or decompress

5.2.2.4 QzHuffmanHdr_T QzSessionParams_S::huffman_hdr

Dynamic or Static Huffman headers

5.2.2.5 unsigned int QzSessionParams_S::hw_buff_sz

default buffer size, Must be a power of 2 4K,8K,16K,32K,64K,128K

5.2.2.6 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

5.2.2.7 unsigned int QzSessionParams_S::max_forks

maximum forks permitted in the current thread. 0 means no forking permitted

5.2.2.8 unsigned int QzSessionParams_S::poll_sleep

<Compress/decompression algorithms nanosleep between poll [0..1000] 0 means no sleep

5.2.2.9 unsigned int QzSessionParams_S::req_cnt_threshold

5.2.2.10 unsigned char QzSessionParams_S::sw_backup

0 means no sw backup, 1 means sw backup

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 stat usof 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 hw 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 properly running 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](#)

Chapter 6

File Documentation

6.1 include/qatzip.h File Reference

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

Classes

- struct [QzSessionParams_S](#)
- struct [QzSession_S](#)
- struct [QzStatus_S](#)

Macros

- #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_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 [QZ_SNAPPY](#) ((unsigned char)'S')
- #define [QZ_LZ4](#) ((unsigned char)'4')
- #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_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_COMP_THRESHOLD_DEFAULT 1024`
- `#define QZ_COMP_THRESHOLD_MINIMUM 128`
- `#define QZ_REQ_THRESHOLD_MINIMUM 1`
- `#define QZ_REQ_THRESHOLD_MAXINUM 4`
- `#define QZ_REQ_THRESHOLD_DEFAULT 4`
- `#define QZ_SKID_PAD_SZ 48`

Typedefs

- `typedef enum QzHuffmanHdr_E QzHuffmanHdr_T`
- `typedef enum QzDirection_E QzDirection_T`
- `typedef struct QzSessionParams_S QzSessionParams_T`
- `typedef struct QzSession_S QzSession_T`
- `typedef struct QzStatus_S QzStatus_T`

Enumerations

- `enum QzHuffmanHdr_E { QZ_DYNAMIC_HDR = 0, QZ_STATIC_HDR }`
- `enum QzDirection_E { QZ_DIR_COMPRESS = 0, QZ_DIR_DECOMPRESS, QZ_DIR_BOTH }`

Functions

- `int qzInit (QzSession_T *sess, unsigned char sw_backup)`
- `int qzSetupSession (QzSession_T *sess, QzSessionParams_T *params)`
- `int qzCompress (QzSession_T *sess, const unsigned char *src, unsigned int *src_len, unsigned char *dest, unsigned int *dest_len, unsigned int last)`
- `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)`
- `int qzDecompress (QzSession_T *sess, const unsigned char *src, unsigned int *src_len, unsigned char *dest, unsigned int *dest_len)`
- `int qzTeardownSession (QzSession_T *sess)`
- `int qzClose (QzSession_T *sess)`
- `int qzGetStatus (QzSession_T *sess, QzStatus_T *status)`
- `unsigned int qzMaxCompressedLength (unsigned int src_sz)`
- `int qzSetDefaults (QzSessionParams_T *defaults)`
- `int qzGetDefaults (QzSessionParams_T *defaults)`
- `void * qzMalloc (size_t sz, int numa, int force_pinned)`
- `void qzFree (void *m)`
- `int qzMemFindAddr (unsigned char *a)`

6.1.1 Macro Definition Documentation

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

6.1.1.2 `#define QZ_BUF_ERROR (-3)`

Insufficient buffer error

6.1.1.3 `#define QZ_COMP_ALGOL_DEFAULT QZ_DEFLATE`

6.1.1.4 `#define QZ_COMP_LEVEL_DEFAULT 1`

6.1.1.5 `#define QZ_COMP_THRESHOLD_DEFAULT 1024`

6.1.1.6 `#define QZ_COMP_THRESHOLD_MINIMUM 128`

6.1.1.7 `#define QZ_DATA_ERROR (-4)`

Input data was corrupted

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

6.1.1.9 `#define QZ_DIRECTION_DEFAULT QZ_DIR_BOTH`

6.1.1.10 `#define QZ_DUPLICATE (1)`

Can not process function again. No failure.

6.1.1.11 `#define QZ_FAIL (-2)`

Unspecified error

6.1.1.12 `#define QZ_FORCE_SW (2)`

using SW: Switch to software because of previous block

6.1.1.13 `#define QZ_HUFF_HDR_DEFAULT QZ_DYNAMIC_HDR`

6.1.1.14 `#define QZ_HW_BUFF_MAX_SZ (512*1024)`

6.1.1.15 `#define QZ_HW_BUFF_MIN_SZ (1*1024)`

6.1.1.16 `#define QZ_HW_BUFF_SZ (64*1024)`

6.1.1.17 `#define QZ_LOW_MEM (14)`

using SW: Not enough pinned memory

6.1.1.18 `#define QZ_LZ4 ((unsigned char)'4')`

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

6.1.1.20 `#define QZ_MAX_FORK_DEFAULT 3`

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

6.1.1.22 `#define QZ_NO_HW (11)`

using SW: No QAT HW detected

6.1.1.23 #define QZ_NO_INST_ATTACH (13)

using SW: Could not attach to an instance

6.1.1.24 #define QZ_NO_MDRV (12)

using SW: No memory driver detected

6.1.1.25 #define QZ_NOSW_LOW_MEM (-104)

not using SW: not enough pinned memory

6.1.1.26 #define QZ_NOSW_NO_HW (-101)

not using SW: No QAT HW detected

6.1.1.27 #define QZ_NOSW_NO_INST_ATTACH (-103)

not using SW: Could not attach to instance

6.1.1.28 #define QZ_NOSW_NO_MDRV (-102)

not using SW: No memory driver detected

6.1.1.29 #define QZ_PARAMS (-1)

invalid parameter in function call

6.1.1.30 #define QZ_POLL_SLEEP_DEFAULT 10

6.1.1.31 #define QZ_REQ_THRESHOLD_DEFAULT 4

6.1.1.32 #define QZ_REQ_THRESHOLD_MAXIMUM 4

6.1.1.33 #define QZ_REQ_THRESHOLD_MINIMUM 1

6.1.1.34 #define QZ_SNAPPY ((unsigned char)'S')

6.1.1.35 #define QZ_SW_BACKUP_DEFAULT 1

6.1.2 Function Documentation

6.1.2.1 unsigned int qzMaxCompressedLength (unsigned int *src_sz*)

Index

- algo_hw
 - QzStatus_S, [23](#)
- algo_sw
 - QzStatus_S, [23](#)
- comp_algorithm
 - QzSessionParams_S, [22](#)
- comp_lvl
 - QzSessionParams_S, [22](#)
- Data Compression API
 - QZ_DIR_BOTH, [10](#)
 - QZ_DIR_COMPRESS, [10](#)
 - QZ_DIR_DECOMPRESS, [10](#)
 - QZ_DYNAMIC_HDR, [11](#)
 - QZ_STATIC_HDR, [11](#)
- Data Compression API, [7](#)
 - QZ_OK, [8](#)
 - QZ_SKID_PAD_SZ, [8](#)
 - qzClose, [11](#)
 - qzCompress, [11](#)
 - qzCompressCrc, [12](#)
 - qzDecompress, [13](#)
 - QzDirection_E, [10](#)
 - QzDirection_T, [9](#)
 - qzFree, [14](#)
 - qzGetDefaults, [14](#)
 - qzGetStatus, [15](#)
 - QzHuffmanHdr_E, [10](#)
 - QzHuffmanHdr_T, [9](#)
 - qzInit, [16](#)
 - qzMalloc, [16](#)
 - qzMemFindAddr, [18](#)
 - QzSession_T, [9](#)
 - QzSessionParams_T, [9](#)
 - qzSetDefaults, [18](#)
 - qzSetupSession, [19](#)
 - QzStatus_T, [9](#)
 - qzTeardownSession, [20](#)
- direction
 - QzSessionParams_S, [22](#)
- huffman_hdr
 - QzSessionParams_S, [22](#)
- hw_buff_sz
 - QzSessionParams_S, [22](#)
- hw_session_stat
 - QzSession_S, [21](#)
- hw_session_status
 - QzStatus_S, [24](#)
- include/qatzip.h, [25](#)
- input_sz_thrshold
 - QzSessionParams_S, [22](#)
- internal
 - QzSession_S, [21](#)
- MIN
 - qatzip.h, [26](#)
- max_forks
 - QzSessionParams_S, [23](#)
- memory_allocated
 - QzStatus_S, [24](#)
- poll_sleep
 - QzSessionParams_S, [23](#)
- QZ_DIR_BOTH
 - Data Compression API, [10](#)
- QZ_DIR_COMPRESS
 - Data Compression API, [10](#)
- QZ_DIR_DECOMPRESS
 - Data Compression API, [10](#)
- QZ_DYNAMIC_HDR
 - Data Compression API, [11](#)
- QZ_STATIC_HDR
 - Data Compression API, [11](#)
- QZ_BUF_ERROR
 - qatzip.h, [26](#)
- QZ_DATA_ERROR
 - qatzip.h, [27](#)
- QZ_DEFLATE
 - qatzip.h, [27](#)
- QZ_DIRECTION_DEFAULT
 - qatzip.h, [27](#)
- QZ_DUPLICATE
 - qatzip.h, [27](#)
- QZ_FAIL
 - qatzip.h, [27](#)
- QZ_FORCE_SW
 - qatzip.h, [27](#)
- QZ_HUFF_HDR_DEFAULT
 - qatzip.h, [27](#)
- QZ_HW_BUFF_MAX_SZ
 - qatzip.h, [27](#)
- QZ_HW_BUFF_MIN_SZ
 - qatzip.h, [27](#)
- QZ_HW_BUFF_SZ
 - qatzip.h, [27](#)
- QZ_LOW_MEM
 - qatzip.h, [27](#)

- QZ_LZ4
 - qatzip.h, [27](#)
- QZ_MAX_ALGORITHMS
 - qatzip.h, [27](#)
- QZ_MAX_FORK_DEFAULT
 - qatzip.h, [27](#)
- QZ_MEMCPY
 - qatzip.h, [27](#)
- QZ_NO_HW
 - qatzip.h, [27](#)
- QZ_NO_INST_ATTACH
 - qatzip.h, [27](#)
- QZ_NO_MDRV
 - qatzip.h, [28](#)
- QZ_NOSW_LOW_MEM
 - qatzip.h, [28](#)
- QZ_NOSW_NO_HW
 - qatzip.h, [28](#)
- QZ_NOSW_NO_MDRV
 - qatzip.h, [28](#)
- QZ_OK
 - Data Compression API, [8](#)
- QZ_PARAMS
 - qatzip.h, [28](#)
- QZ_SKID_PAD_SZ
 - Data Compression API, [8](#)
- QZ_SNAPPY
 - qatzip.h, [28](#)
- qat_hw_count
 - QzStatus_S, [24](#)
- qat_instance_attach
 - QzStatus_S, [24](#)
- qat_mem_drvr
 - QzStatus_S, [24](#)
- qat_service_stated
 - QzStatus_S, [24](#)
- qatzip.h
 - MIN, [26](#)
 - QZ_BUF_ERROR, [26](#)
 - QZ_DATA_ERROR, [27](#)
 - QZ_DEFLATE, [27](#)
 - QZ_DUPLICATE, [27](#)
 - QZ_FAIL, [27](#)
 - QZ_FORCE_SW, [27](#)
 - QZ_HW_BUFF_MAX_SZ, [27](#)
 - QZ_HW_BUFF_MIN_SZ, [27](#)
 - QZ_HW_BUFF_SZ, [27](#)
 - QZ_LOW_MEM, [27](#)
 - QZ_LZ4, [27](#)
 - QZ_MAX_ALGORITHMS, [27](#)
 - QZ_MEMCPY, [27](#)
 - QZ_NO_HW, [27](#)
 - QZ_NO_INST_ATTACH, [27](#)
 - QZ_NO_MDRV, [28](#)
 - QZ_NOSW_LOW_MEM, [28](#)
 - QZ_NOSW_NO_HW, [28](#)
 - QZ_NOSW_NO_MDRV, [28](#)
 - QZ_PARAMS, [28](#)
 - QZ_SNAPPY, [28](#)
 - qzMaxCompressedLength, [28](#)
- qzClose
 - Data Compression API, [11](#)
- qzCompress
 - Data Compression API, [11](#)
- qzCompressCrc
 - Data Compression API, [12](#)
- qzDecompress
 - Data Compression API, [13](#)
- QzDirection_E
 - Data Compression API, [10](#)
- QzDirection_T
 - Data Compression API, [9](#)
- qzFree
 - Data Compression API, [14](#)
- qzGetDefaults
 - Data Compression API, [14](#)
- qzGetStatus
 - Data Compression API, [15](#)
- QzHuffmanHdr_E
 - Data Compression API, [10](#)
- QzHuffmanHdr_T
 - Data Compression API, [9](#)
- qzInit
 - Data Compression API, [16](#)
- qzMalloc
 - Data Compression API, [16](#)
- qzMaxCompressedLength
 - qatzip.h, [28](#)
- qzMemFindAddr
 - Data Compression API, [18](#)
- QzSession_S, [21](#)
 - hw_session_stat, [21](#)
 - internal, [21](#)
 - thd_sess_stat, [21](#)
 - total_in, [21](#)
 - total_out, [21](#)
- QzSession_T
 - Data Compression API, [9](#)
- QzSessionParams_S, [22](#)
 - comp_algorithm, [22](#)
 - comp_lvl, [22](#)
 - direction, [22](#)
 - huffman_hdr, [22](#)
 - hw_buff_sz, [22](#)
 - input_sz_thrshold, [22](#)
 - max_forks, [23](#)
 - poll_sleep, [23](#)
 - req_cnt_thrshold, [23](#)
 - sw_backup, [23](#)
- QzSessionParams_T
 - Data Compression API, [9](#)
- qzSetDefaults
 - Data Compression API, [18](#)
- qzSetupSession
 - Data Compression API, [19](#)
- QzStatus_S, [23](#)

- algo_hw, [23](#)
- algo_sw, [23](#)
- hw_session_status, [24](#)
- memory_allocated, [24](#)
- qat_hw_count, [24](#)
- qat_instance_attach, [24](#)
- qat_mem_drvr, [24](#)
- qat_service_stated, [24](#)
- using_huge_pages, [24](#)
- QzStatus_T
 - Data Compression API, [9](#)
- qzTeardownSession
 - Data Compression API, [20](#)
- req_cnt_thrshold
 - QzSessionParams_S, [23](#)
- sw_backup
 - QzSessionParams_S, [23](#)
- thd_sess_stat
 - QzSession_S, [21](#)
- total_in
 - QzSession_S, [21](#)
- total_out
 - QzSession_S, [21](#)
- using_huge_pages
 - QzStatus_S, [24](#)