QATzip

1.0.1

Generated by Doxygen 1.8.5

Wed Oct 9 2019 05:47:28

Contents

Chapter 1

Module Index

1.1 Modules

Here is a list of all modules:

Chapter 2

Class Index

2.1 Class List

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

Chapter 3

File Index

3.1 File List

Here is a list of all files with brief descriptions:

Chapter 4

Module Documentation

4.1 Data Compression API

Classes

- struct QzSessionParams_S
- struct QzSession_S
- struct QzStatus_S
- struct QzStream_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 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_T { COMMON_MEM = 0, PINNED_MEM }
- enum QzDirection_E { QZ_DIR_COMPRESS = 0, QZ_DIR_DECOMPRESS, QZ_DIR_BOTH }
- enum QzDataFormat_E { QZ_DEFLATE_RAW = 0, QZ_DEFLATE_GZIP, QZ_DEFLATE_GZIP_EXT, QZ_-FMT_NUM }
- enum QzCrcType_E { QZ_CRC32 = 0, QZ_CRC64, QZ_ADLER, NONE }

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)
- int qzCompressStream (QzSession_T ∗sess, QzStream_T ∗strm, unsigned int last)
- int qzDecompressStream (QzSession_T ∗sess, QzStream_T ∗strm, unsigned int last)
- 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 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

in	*src_sz*	Input data length in byte.

Return values

dest_sz	Max compressed data output length in byte. When src_sz equal to 0, the return value is QZ_COMPRESSED_SZ_OF_EMPTY_FILE(34). 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 QzCrcType_E QzCrcType_T

```
Supported checksum type
```

This enumerated list identifies the checksum type for input/output data. A format can be CRC32, CRC64, Adler or none.

4.1.3.2 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.

4.1.3.3 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.

4.1.3.4 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 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 cleanup until the application exits.

```
Supported Huffman Headers
```

This enumerated list identifies the Huffman header types supported by QATZip

4.1.3.5 typedef struct QzSession_S QzSession_T

QATZIP Session opaque data storage

This structure contains a pointer to a structure with session state

4.1.3.6 typedef struct QzSessionParams_S QzSessionParams_T

QATZIP Session Initialization parameters

This structure contains data for initializing a session

4.1.3.7 typedef struct QzStatus_S QzStatus_T

QATZIP status structure

This structure contains data relating to the stat usof QAT on the platform

4.1.3.8 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_T

```
Supported memory types
```

This enumerated list identifies memory types supported by QATZip.

Enumerator

COMMON_MEM	Allocate non-continous memory
PINNED_MEM	Allocate continous memory

4.1.4.2 enum QzCrcType_E

```
Supported checksum type
```

This enumerated list identifies the checksum type for input/output data. A format can be CRC32, CRC64, Adler or none.

Enumerator

QZ_CRC32 CRC32 checksum

QZ_CRC64 CRC64 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_RAW Data is in raw deflate format

QZ_DEFLATE_GZIP Data is in deflate wrappped by GZip header and footer

QZ_DEFLATE_GZIP_EXT Data is in deflate warpped by GZip extension header and footer

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 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, NU-LL); 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 cleanup 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 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`	*sess*	Session handle
`in`	*src*	point to source buffer
`in,out`	*src_len*	length of source buffer. Modified to number of bytes consumed
`in`	*dest*	point to destination buffer
`in,out`	*dest_len*	length of destination buffer. Modified to length of compressed data when function returns
`in`	*last*	1 for 'No more data to be compressed' 0 for 'More data to be compressed'

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

4.1.5.3 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 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 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`	*sess*	Session handle
`in`	*src*	point to source buffer
`in,out`	*src_len*	length of source buffer. Modified to number of bytes consumed
`in`	*dest*	point to destination buffer
`in,out`	*dest_len*	length of destination buffer. Modified to length of compressed data when function returns
`in`	*last*	1 for 'No more data to be compressed' 0 for 'More data to be compressed'
`in,out`	*crc*	point to CRC32 checksum buffer

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

4.1.5.4 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_buf_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/crc64 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
`in,out`	*strm*	Stream handle
`in`	*last*	1 for 'No more data to be compressed' 0 for 'More data to be compressed'

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

4.1.5.5 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 RFC1952.

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

Parameters

`in`	*sess*	Session handle
`in`	*src*	point to source buffer
`in`	*src_len*	length of source buffer. Modified to length of processed compressed data when function returns
`in`	*dest*	point to destination buffer
`in,out`	*dest_len*	length of destination buffer. Modified to length of decompressed data when function returns

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

4.1.5.6 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_buf_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 uncompression blocks in the ∗out of QzStream_T structure and put checksum for uncompressed data in crc32/crc64 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
`in,out`	*strm*	Stream handle
`in`	*last*	1 for 'No more data to be compressed' 0 for 'More data to be compressed'

Return values

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

4.1.5.8 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

in	*m*	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 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

in	*defaults*	The pointer to default value.

Return values

QZ_OK	Success on getting default value.
QZ_PARAM	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 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_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_NOS-W_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*	pointer to opaque instance and session data.
`in`	*status*	pointer to QATZIP status structure.

Return values

QZ_OK	Function executed successfully. A 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 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.

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 hardware 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 hardware or sw instance has been allocated to the calling process/thread.
QZ_DUPLICATE	This process/thread already has a hardware 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

4.1.5.12 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	*sz*	Memory size to be allocated.
in	*numa*	NUMA node from which to allocate memory
in	*force_pinned*	PINNED_MEM allocate continous memory COMMON_MEM allocate non-continous memory

Return values

NULL	Fail to allocate memory
adress	The address to allocated memory

Precondition

None

Postcondition

None

Note

Only a synchronous version of this function is provided.

See Also

None

4.1.5.13 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 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

4.1.5.15 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 hardware or sw based compression session has been created.

QZ_PARAMS	∗sess is NULL or member of params is invalid
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_NO_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

4.1.5.16 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

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

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..9

5.2.2.3 QzDataFormat_T QzSessionParams_S::data_fmt

defalte, 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 2 4K,8K,16K,32K,64K,128K

5.2.2.7 unsigned int QzSessionParams_S::input_sz_thrshold

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_thrshold

set between 1 and 4, default 4

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_thrshold

when previous try failed, wait for specific number of call before retry device open. 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_alloced
- 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 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_alloced

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

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 crc_64
- 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 unsigned long long QzStream_S::crc_64

Checksum value for 64bit CRC

5.4.2.3 QzCrcType_T QzStream_S::crc_type

Checksum type in Adler, CRC32, CRC64 or none

5.4.2.4 unsigned char∗ QzStream_S::in

Input data pointer set by application

5.4.2.5 unsigned int QzStream_S::in_sz

Set by application, reset by QATZip to indicate consumed data

5.4.2.6 void∗ QzStream_S::opaque

Internal storage managed by QATZip

5.4.2.7 unsigned char∗ QzStream_S::out

Output data pointer set by application

5.4.2.8 unsigned int QzStream_S::out_sz

Set by application, reset by QATZip to indicate processed data

5.4.2.9 unsigned int QzStream_S::pending_in

Unprocessed bytes held in QATZip

5.4.2.10 unsigned int QzStream_S::pending_out

Processed bytes held in QATZip

5.4.2.11 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 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 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_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_MAXINUM NUM_BUFF
- #define QZ_REQ_THRESHOLD_DEFAULT QZ_REQ_THRESHOLD_MAXINUM
- #define QZ_WAIT_CNT_THRESHOLD_DEFAULT 8
- #define QZ_SKID_PAD_SZ 48
- #define QZ_COMPRESSED_SZ_OF_EMPTY_FILE 34

Typedefs

- typedef enum QzHuffmanHdr_E QzHuffmanHdr_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_T { COMMON_MEM = 0, PINNED_MEM }
- enum QzDirection_E { QZ_DIR_COMPRESS = 0, QZ_DIR_DECOMPRESS, QZ_DIR_BOTH }
- enum QzDataFormat_E { QZ_DEFLATE_RAW = 0, QZ_DEFLATE_GZIP, QZ_DEFLATE_GZIP_EXT, QZ_-FMT_NUM }
- enum QzCrcType_E { QZ_CRC32 = 0, QZ_CRC64, QZ_ADLER, NONE }

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)
- int qzCompressStream (QzSession_T ∗sess, QzStream_T ∗strm, unsigned int last)
- int qzDecompressStream (QzSession_T ∗sess, QzStream_T ∗strm, unsigned int last)
- 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 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_COMPRESSED_SZ_OF_EMPTY_FILE 34

6.1.1.8 #define QZ_DATA_ERROR (-4)

Input data was corrupted

6.1.1.9 #define QZ_DATA_FORMAT_DEFAULT QZ_DEFLATE_GZIP_EXT

6.1.1.10 #define QZ_DEFLATE ((unsigned char)8)

6.1.1.11 #define QZ_DIRECTION_DEFAULT QZ_DIR_BOTH

6.1.1.12 #define QZ_DUPLICATE (1)

Can not process function again. No failure.

6.1.1.13 #define QZ_FAIL (-2)

Unspecified error

6.1.1.14 #define QZ_FORCE_SW (2)

using SW: Switch to software because of previous block

6.1.1.15 #define QZ_HUFF_HDR_DEFAULT QZ_DYNAMIC_HDR

6.1.1.16 #define QZ_HW_BUFF_MAX_SZ (512∗1024)

6.1.1.17 #define QZ_HW_BUFF_MIN_SZ (1∗1024)

6.1.1.18 #define QZ_HW_BUFF_SZ (64∗1024)

6.1.1.19 #define QZ_LOW_DEST_MEM (15)

using SW: Not enough pinned memory for dest buffer

6.1.1.20 #define QZ_LOW_MEM (14)

using SW: Not enough pinned memory

6.1.1.21 #define QZ_LZ4 ((unsigned char)'4')

6.1.1.22 #define QZ_MAX_ALGORITHMS ((int)255)

6.1.1.23 #define QZ_MAX_FORK_DEFAULT 3

6.1.1.24 #define QZ_MEMCPY(*dest, src, dest_sz, src_sz*) memcpy((void ∗)(dest), (void ∗) (src), (size_t)MIN(dest_sz, src_sz))

6.1.1.25 #define QZ_NO_HW (11)

using SW: No QAT HW detected

6.1.1.26 #define QZ_NO_INST_ATTACH (13)

using SW: Could not attach to an instance

6.1.1.27 #define QZ_NO_MDRV (12)

using SW: No memory driver detected

6.1.1.28 #define QZ_NONE (100)

device uninitialzied

6.1.1.29 #define QZ_NOSW_LOW_MEM (-104)

not using SW: not enough pinned memory

6.1.1.30 #define QZ_NOSW_NO_HW (-101)

not using SW: No QAT HW detected

6.1.1.31 #define QZ_NOSW_NO_INST_ATTACH (-103)

not using SW: Could not attach to instance

6.1.1.32 #define QZ_NOSW_NO_MDRV (-102)

not using SW: No memory driver detected

6.1.1.33 #define QZ_PARAMS (-1)

invalid parameter in function call

6.1.1.34 #define QZ_POLL_SLEEP_DEFAULT 10

6.1.1.35 #define QZ_REQ_THRESHOLD_DEFAULT QZ_REQ_THRESHOLD_MAXINUM

6.1.1.36 #define QZ_REQ_THRESHOLD_MAXINUM NUM_BUFF

6.1.1.37 #define QZ_REQ_THRESHOLD_MINIMUM 1

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

6.1.1.39 #define QZ_STRM_BUFF_MAX_SZ (2∗1024∗1024 - 5∗1024)

6.1.1.40 #define QZ_STRM_BUFF_MIN_SZ (1∗1024)

6.1.1.41 #define QZ_STRM_BUFF_SZ_DEFAULT QZ_HW_BUFF_SZ

6.1.1.42 #define QZ_SW_BACKUP_DEFAULT 1

6.1.1.43 #define QZ_WAIT_CNT_THRESHOLD_DEFAULT 8

6.1.2 Function Documentation

6.1.2.1 unsigned int qzMaxCompressedLength (unsigned int *src_sz*)

Index