QATzip

1.0.8

Generated by Doxygen 1.8.13

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 QATZIP_API_VERSION_NUM_MAJOR (2)
- #define QATZIP_API_VERSION_NUM_MINOR (2)
- #define QZ_OK (0)
- #define QZ_SW_EXECUTION_BIT (4)
- #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 int(∗ qzCallbackFn) (void ∗external, const unsigned char ∗src, unsigned int ∗src_len, unsigned char ∗dest, unsigned int ∗dest_len, int ∗ExtStatus)
- 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 { UNSPECIFIED = 0, COMMON_MEM = 0, PINNED_MEM, SV_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_LZ4_FH, QZ_LZ4S_FH, QZ_LZ4S_PP, QZ_ZSTD_RAW }
- enum QzCrcType_E { NONE = 0, QZ_CRC32, QZ_ADLER }

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 QATZIP_API_VERSION_NUM_MAJOR

```
#define QATZIP_API_VERSION_NUM_MAJOR (2)
```

QATzip Major Version Number The QATzip API major version number. This number will be incremented when significant changes to the API have occurred. The combination of the major and minor number definitions represent the complete version number for this interface.

4.1.2.2 QATZIP_API_VERSION_NUM_MINOR

```
#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 have occurred. The combination of the major and minor number definitions represent the complete version number for this interface.

4.1.2.3 QZ_OK

```
#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 QZ_SKID_PAD_SZ

```
#define QZ_SKID_PAD_SZ 48
```

Get the maximum compressed output length

Get the maximum 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 bytes sess Session handle (pointer to opaque instance and session data)

Return values

dest_sz	Max compressed data output length in bytes. When src_sz is 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

4.1.2.5 QZ_SW_EXECUTION_BIT

```
#define QZ_SW_EXECUTION_BIT (4)
```

QATzip Extended return information

The following definitions can be used with the extended return values.

QZ_SW_EXECUTION indicates if a request for services was performed in software.

QZ_HW_TIMEOUT indicates if a request to hardware was timed out.

4.1.3 Typedef Documentation

4.1.3.1 PinMem_T

```
typedef enum PinMem_E PinMem_T
```

Supported memory types

This enumerated list identifies memory types supported by QATzip.

4.1.3.2 qzCallbackFn

```
typedef int(* qzCallbackFn) (void *external, const unsigned char *src, unsigned int *src_len,
unsigned char *dest, unsigned int *dest_len, int *ExtStatus)
```

Post processing callback after LZ4s compression

This function will be called in qzCompressCrc for post processing of lz4s payloads. Function implementation should be provided by user and comply with this prototype's rules. The input paramter 'dest' will contain the compressed lz4s format data.

The user callback function should be provided through the QzSessionParams_T. And set data format of compression to 'QZ_LZ4S_FH', then post-processing will be trigger.

qzCallback's first parameter 'external' can be a customized compression context which can be setup before QAT qzSetupSession. It can be provided to QATZip though the 'qzCallback_external' variable in the QzSessionParams_T structure.

ExtStatus will be embedded into extended return codes when qzCallbackFn return `QZ_POST_PROCESS_ERROR`. See extended return code section and *Ext API for details.

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

Parameters

in	*external*	User context provided through the 'qzCallback_external' pointer in the QzSessionParams_T structure.
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,out	*ExtStatus*	'qzCallback' customized error code

Return values

QZ_OK	Function executed successfully
QZ_FAIL	Function did not succeed
QZ_PARAMS	params are invalid
QZ_POST_PROCESS_ERROR	post processing error

Precondition

None

Postcondition

None

Note

Only a synchronous version of this function is provided.

See also

None

4.1.3.3 QzCrcType_T

```
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.

4.1.3.4 QzDataFormat_T

```
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.

4.1.3.5 QzDirection_T

```
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.

4.1.3.6 QzHuffmanHdr_T

```
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 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, 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, N↩ ULL); 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.

Additions for QAT 2.0 and beyond platforms.

1. Addition of LZ4 and LZ4s
2. Addition of post processing functions for out of LZ4s
    - support for zstd compress via a pre-canned post processing function
    - support for zstd decompress via software only
3. Compression level up to 12 for LZ4 and LZ4s
4. Support for Shared Virtual Memory
    - default off for QAT 1.x sessions
    - default on for QAT 2.0 sessions and beyond
5. Support for gzip header with additional compression algorithms

```
Supported Huffman Headers
```

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

4.1.3.7 QzSession_T

```
typedef struct QzSession_S QzSession_T
```

QATzip Session opaque data storage

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

4.1.3.8 QzSessionParams_T

```
typedef struct QzSessionParams_S QzSessionParams_T
```

QATzip Session Initialization parameters

This structure contains data for initializing a session.

4.1.3.9 QzStatus_T

```
typedef struct QzStatus_S QzStatus_T
```

QATzip status structure

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

4.1.3.10 QzStream_T

```
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 PinMem_E

```
enum PinMem_E
```

Supported memory types

This enumerated list identifies memory types supported by QATzip.

Enumerator

UNSPECIFIED	Type of memory is not specified
COMMON_MEM	Allocate non-contiguous memory
PINNED_MEM	Allocate contiguous memory
SV_MEM	Shared Virtual Memory will be used

4.1.4.2 QzCrcType_E

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

NONE	No checksum
QZ_CRC32	CRC32 checksum
QZ_ADLER	Adler checksum

4.1.4.3 QzDataFormat_E

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 extended header and footer
QZ_DEFLATE_RAW	Data is in raw deflate format
QZ_LZ4_FH	Data is in LZ4 format with frame headers
QZ_LZ4S_FH	Data is in LZ4s format with frame headers
QZ_LZ4S_PP	Data is in LZ4s format and has been post processed
QZ_ZSTD_RAW	Data is in raw zStandard format

4.1.4.4 QzDirection_E

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 QzHuffmanHdr_E

`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 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, 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, N↩ ULL); 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.

Additions for QAT 2.0 and beyond platforms.

1. Addition of LZ4 and LZ4s
2. Addition of post processing functions for out of LZ4s
    - support for zstd compress via a pre-canned post processing function

- support for zstd decompress via software only

3. Compression level up to 12 for LZ4 and LZ4s
4. Support for Shared Virtual Memory
   - default off for QAT 1.x sessions
   - default on for QAT 2.0 sessions and beyond
5. Support for gzip header with additional compression algorithms

```
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 qzClose()

```
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 qzCompress()

```
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, as 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 (pointer to opaque instance and session data)
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	*ext_rc*	qzCompressExt only. If not NULL, ext_rc point to a location where extended return codes may be returned. See extended return code section for details. if NULL, no extended information will be provided.

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 qzCompressCrc()

```
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, as 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	*sess*	Session handle (pointer to opaque instance and session data)
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*	Pointer to CRC32 checksum buffer
in,out	*ext_rc*	qzCompressCrcExt only. If not NULL, ext_rc point to a location where extended return codes may be returned. See extended return code section for details. if NULL, no extended information will be provided.

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 qzCompressStream()

```
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

`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' (always set to 1 in the Microsoft(R) Windows(TM) QATzip implementation)

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 qzDecompress()

```
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, as per RFC 1952.

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	*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
in,out	*ext_rc*	qzDecompressExt only. If not NULL, ext_rc point to a location where extended return codes may be returned. See extended return code section for details. if NULL, no extended information will be provided.

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 qzDecompressStream()

```
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

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 qzEndStream()

```
QATZIP_API int qzEndStream (
            QzSession_T * sess,
            QzStream_T * strm )
```

Terminates a QATzip stream

This function disconnects 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

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 qzFree()

```
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

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 qzGetDefaults()

```
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

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 qzGetStatus()

```
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_init 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_status 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_NO_MDRV QZ_NOSW_NO_INST_ATTACH QZ_NOSW_LOW_MEM QZ_NO_SW_AVAIL

Applications should verify the elements of the status structure are correct for the required operations. It should be noted that some information will be available only after qzInit has been called, either implicitly or explicitly. The qat_service_init element of the status structure will indicate if initialization has taken place.

The hw_session_status will depend on the availability of hardware based compression and software based compression. The following table indicates what hw_session_status based on the availability of compression engines and the sw_backup flag.

| HW | SW Engine | sw_backup | hw_session_stat |

avail	avail	setting	
N	N	0	QZ_NOSW_NO_HW
N	N	1	QZ_NOSW_NO_HW
N	Y	0	QZ_FAIL
N	Y	1	QZ_NO_HW (1)
Y	N	0	QZ_OK
Y	N	1	QZ_NO_SW_AVAIL (2)
Y	Y	0	QZ_OK
Y	Y	1	QZ_OK

Note 1: If an application indicates software backup is required by setting sw_backup=1, and a software engine is available and if no hardware based compression engine is available then the hw_session_status will be set to QZ_NO_HW. All compression and decompression will use the software engine. Note 2: If an application indicates software backup is required by setting sw_backup=1, and if no software based compression engine is available then the hw_session_status will be set to QZ_NO_SW_AVAIL. In this case, QAT based compression may be used however no software backup will available. If the application relies on software backup being avialable, then this return code can be treated as an error. 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 qzInit()

```
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	*sess*	Session handle (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 software 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 software session being established
QZ_NOSW_NO_MDRV	No memory driver. No software session established
QZ_NOSW_NO_INST_ATTACH	No instance available No software session established
QZ_NOSW_LOW_MEM	Not enough pinned memory available No software session established
QZ_UNSUPPORTED_FMT	No support for requested algorithm; using software
QZ_NOSW_UNSUPPORTED_FMT	No support for requested algorithm; No software session established
QZ_NO_SW_AVAIL	No software is available. This will be returned when sw_backup is set to 1 but the session does not support software backup or software backup is unavailable to the application.

Precondition

None

Postcondition

None

Note

Only a synchronous version of this function is provided.

See also

None

4.1.5.12 qzMalloc()

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

Return values

NULL	Fail to allocate memory
address	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 qzMemFindAddr()

```
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 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 qzSetDefaults()

```
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

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 qzSetupSession()

```
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 QZ_DUPLICATE will be returned without modifying the existing session.

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	*params*	Parameters for session

Return values

QZ_OK	Function executed successfully. A hardware or software based compression session has been created
QZ_DUPLICATE	∗sess includes an existing hardware or software session
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_ATTACH	No instance available No software session established
QZ_NO_LOW_MEM	Not enough pinned memory available No software session established
QZ_UNSUPPORTED_FMT	No support for requested algorithm; using software
QZ_NOSW_UNSUPPORTED_FMT	No support for requested algorithm; No software session established
QZ_NO_SW_AVAIL	No software is available. This may returned when sw_backup is set to 1 but the session does not support software backup or software backup is unavailable to the application.

Precondition

None

Postcondition

None

Note

Only a synchronous version of this function is provided.

See also

None

4.1.5.16 qzTeardownSession()

```
QATZIP_API int qzTeardownSession (
            QzSession_T * sess )
```

Uninitialize 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 (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

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 hw_session_stat

`signed long int QzSession_S::hw_session_stat`

Filled in during initialization, session startup and decompression

5.1.2.2 internal

```
void* QzSession_S::internal
```

Session data is opaque to outside world

5.1.2.3 thd_sess_stat

```
int QzSession_S::thd_sess_stat
```

Note process compression and decompression thread state

5.1.2.4 total_in

```
unsigned long QzSession_S::total_in
```

Total processed input data length in this session

5.1.2.5 total_out

```
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
- PinMem_T mem_type
- qzCallbackFn qzCallback
- void * qzCallback_external
- unsigned int is_busy_polling
- unsigned int is_sensitive_mode
- unsigned int lz4s_mini_match

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 comp_algorithm

```
unsigned char QzSessionParams_S::comp_algorithm
```

Compress/decompression algorithms

5.2.2.2 comp_lvl

```
unsigned int QzSessionParams_S::comp_lvl
```

Compression level 1 to 12. If the comp_algorithm is deflate, values $>$ 12 will be set to 12

5.2.2.3 data_fmt

```
QzDataFormat_T QzSessionParams_S::data_fmt
```

Deflate, deflate with GZip or deflate with GZip ext

5.2.2.4 direction

```
QzDirection_T QzSessionParams_S::direction
```

Compress or decompress

5.2.2.5 huffman_hdr

```
QzHuffmanHdr_T QzSessionParams_S::huffman_hdr
```

If algorithm is deflate, dynamic or Static Huffman headers

5.2.2.6 hw_buff_sz

```
unsigned int QzSessionParams_S::hw_buff_sz
```

Default buffer size For optimal page performance, this value should be a multiple of the page size.

5.2.2.7 input_sz_thrshold

```
unsigned int QzSessionParams_S::input_sz_thrshold
```

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

5.2.2.8 is_busy_polling

```
unsigned int QzSessionParams_S::is_busy_polling
```

0 means no busy polling, 1 means busy polling

5.2.2.9 is_sensitive_mode

```
unsigned int QzSessionParams_S::is_sensitive_mode
```

0 means disable sensitive mode, 1 means enable sensitive mode

5.2.2.10 lz4s_mini_match

```
unsigned int QzSessionParams_S::lz4s_mini_match
```

Set lz4s dictionary mini match, which would be 3 or 4

5.2.2.11 max_forks

```
unsigned int QzSessionParams_S::max_forks
```

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

5.2.2.12 mem_type

```
PinMem_T QzSessionParams_S::mem_type
```

If not specified, default will be Pinned for qat 1.x and common for QAT 2.0

5.2.2.13 qzCallback

```
qzCallbackFn QzSessionParams_S::qzCallback
```

post processing callback for zstd compression

5.2.2.14 qzCallback_external

```
void* QzSessionParams_S::qzCallback_external
```

An opaque pointer provided by the user to be passed into qzCallback during post processing

5.2.2.15 req_cnt_thrshold

```
unsigned int QzSessionParams_S::req_cnt_thrshold
```

Set between 1 and NUM_BUFF, default NUM_BUFF NUM_BUFF is defined in qatzip_internal.h

5.2.2.16 strm_buff_sz

```
unsigned int QzSessionParams_S::strm_buff_sz
```

Stream buffer size between [1K .. 2M - 5K] Default strm_buf_sz equals to hw_buff_sz

5.2.2.17 sw_backup

```
unsigned char QzSessionParams_S::sw_backup
```

0 means no sw backup, 1 means sw backup

5.2.2.18 wait_cnt_thrshold

```
unsigned int QzSessionParams_S::wait_cnt_thrshold
```

When previous try failed, wait for specific number of calls before retrying 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_init
- 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_comp [QZ_MAX_ALGORITHMS]
- unsigned char algo_hw_comp [QZ_MAX_ALGORITHMS]
- unsigned char algo_sw_decomp [QZ_MAX_ALGORITHMS]
- unsigned char algo_hw_decomp [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 algo_hw_comp

```
unsigned char QzStatus_S::algo_hw_comp[QZ_MAX_ALGORITHMS]
```

Count of hardware devices supporting algorithms for compression

5.3.2.2 algo_hw_decomp

```
unsigned char QzStatus_S::algo_hw_decomp[QZ_MAX_ALGORITHMS]
```

Count of hardware devices supporting algorithms for decompression

5.3.2.3 algo_sw_comp

```
unsigned char QzStatus_S::algo_sw_comp[QZ_MAX_ALGORITHMS]
```

Support software algorithms for compression

5.3.2.4 algo_sw_decomp

```
unsigned char QzStatus_S::algo_sw_decomp[QZ_MAX_ALGORITHMS]
```

Support software algorithms for decompresson

5.3.2.5 hw_session_status

```
signed long int QzStatus_S::hw_session_status
```

One of QATzip Session Status

5.3.2.6 memory_alloced

```
unsigned long int QzStatus_S::memory_alloced
```

Amount of memory allocated by this thread/process

5.3.2.7 qat_hw_count

```
unsigned short int QzStatus_S::qat_hw_count
```

From PCI scan

5.3.2.8 qat_instance_attach

```
unsigned char QzStatus_S::qat_instance_attach
```

Is this thread/g_process properly attached to an Instance?

5.3.2.9 qat_mem_drvr

```
unsigned char QzStatus_S::qat_mem_drvr
```

1 if memory driver for QAT exists 2 if memory driver for QAT has been opened 3 memory driver not required. Using SVM. 0 otherwise

5.3.2.10 qat_service_init

```
unsigned char QzStatus_S::qat_service_init
```

Check if the available services have been initialized

5.3.2.11 using_huge_pages

```
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 crc_32

```
unsigned int QzStream_S::crc_32
```

Checksum value

5.4.2.2 crc_type

```
QzCrcType_T QzStream_S::crc_type
```

Checksum type in Adler, CRC32 or none

5.4.2.3 in

```
unsigned char* QzStream_S::in
```

Input data pointer set by application

5.4.2.4 in_sz

```
unsigned int QzStream_S::in_sz
```

Set by application, reset by QATzip to indicate consumed data

5.4.2.5 opaque

```
void* QzStream_S::opaque
```

Internal storage managed by QATzip

5.4.2.6 out

```
unsigned char* QzStream_S::out
```

Output data pointer set by application

5.4.2.7 out_sz

```
unsigned int QzStream_S::out_sz
```

Set by application, reset by QATzip to indicate processed data

5.4.2.8 pending_in

```
unsigned int QzStream_S::pending_in
```

Unprocessed bytes held in QATzip

5.4.2.9 pending_out

```
unsigned int QzStream_S::pending_out
```

Processed bytes held in QATzip

5.4.2.10 reserved

```
unsigned long long QzStream_S::reserved
```

Reserved for future use

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>
#include <stdint.h>
#include <stdbool.h>
```

Classes

- struct QzSessionParams_S
- struct QzSession_S
- struct QzStatus_S
- struct QzStream_S

Macros

- #define QATZIP_API_VERSION_NUM_MAJOR (2)
- #define QATZIP_API_VERSION_NUM_MINOR (2)
- #define QATZIP_API_VERSION
- #define QATZIP_API
- #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_TIMEOUT (-5)
- #define QZ_INTEG (-100)
- #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_UNSUPPORTED_FMT (16)

- #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_NO_SW_AVAIL (-105)
- #define QZ_NOSW_UNSUPPORTED_FMT (-116)
- #define QZ_POST_PROCESS_ERROR (-117)
- #define QZ_MAX_ALGORITHMS ((int)255)
- #define QZ_DEFLATE ((unsigned char)8)
- #define QZ_LZ4 ((unsigned char)'4')
- #define QZ_LZ4s ((unsigned char)'s')
- #define QZ_ZSTD ((unsigned char)'Z')
- #define MIN(a, b) (((a)<(b))?(a):(b))
- #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_SZ_SPR (1∗1024∗1024)
- #define QZ_HW_BUFF_MIN_SZ (1∗1024)
- #define QZ_HW_BUFF_MAX_SZ (512∗1024)
- #define QZ_HW_BUFF_MAX_SZ_SPR (2∗1024∗1024∗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 (12)
- #define QZ_LZS_COMP_LVL_MINIMUM (1)
- #define QZ_LZS_COMP_LVL_MAXIMUM (12)
- #define QZ_PERIODICAL_POLLING (false)
- #define QZ_BUSY_POLLING (true)
- #define QZ_SW_EXECUTION_BIT (4)
- #define QZ_SW_EXECUTION_MASK (1 << QZ_SW_EXECUTION_BIT)
- #define QZ_SW_EXECUTION(ret, ext_rc) (!ret && (ext_rc & QZ_SW_EXECUTION_MASK))
- #define QZ_TIMEOUT_BIT (8)
- #define QZ_TIMEOUT_MASK (1 << QZ_TIMEOUT_BIT)
- #define QZ_HW_TIMEOUT(ret, ext_rc) (!ret && (ext_rc & QZ_TIMEOUT_MASK))
- #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 int(∗ qzCallbackFn) (void ∗external, const unsigned char ∗src, unsigned int ∗src_len, unsigned char ∗dest, unsigned int ∗dest_len, int ∗ExtStatus)
- 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 { UNSPECIFIED = 0, COMMON_MEM = 0, PINNED_MEM, SV_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_LZ4_FH, QZ_LZ4S_FH, QZ_LZ4S_PP, QZ_ZSTD_RAW }
- enum QzCrcType_E { NONE = 0, QZ_CRC32, QZ_ADLER }

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 qzCompressExt (QzSession_T ∗sess, const unsigned char ∗src, unsigned int ∗src_len, unsigned char ∗dest, unsigned int ∗dest_len, unsigned int last, uint64_t ∗ext_rc)
- 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 qzCompressCrcExt (QzSession_T ∗sess, const unsigned char ∗src, unsigned int ∗src_len, unsigned char ∗dest, unsigned int ∗dest_len, unsigned int last, unsigned long ∗crc, uint64_t ∗ext_rc)
- 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 qzDecompressExt (QzSession_T ∗sess, const unsigned char ∗src, unsigned int ∗src_len, unsigned char ∗dest, unsigned int ∗dest_len, uint64_t ∗ext_rc)
- 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 MIN

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

6.1.1.2 QATZIP_API

```
#define QATZIP_API
```

These macros define how the project will be built QATZIP_LINK_DLL must be defined if linking the DLL QATZI↩ P_BUILD_DLL must be defined when building a DLL No definition required if building the project as static library

6.1.1.3 QATZIP_API_VERSION

```
#define QATZIP_API_VERSION
```

Value:

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

6.1.1.4 QZ_BUF_ERROR

```
#define QZ_BUF_ERROR (-3)
```

Insufficient buffer error

6.1.1.5 QZ_BUSY_POLLING

```
#define QZ_BUSY_POLLING (true)
```

6.1.1.6 QZ_COMP_ALGOL_DEFAULT

```
#define QZ_COMP_ALGOL_DEFAULT QZ_DEFLATE
```

6.1.1.7 QZ_COMP_LEVEL_DEFAULT

```
#define QZ_COMP_LEVEL_DEFAULT 1
```

6.1.1.8 QZ_COMP_THRESHOLD_DEFAULT

```
#define QZ_COMP_THRESHOLD_DEFAULT 1024
```

6.1.1.9 QZ_COMP_THRESHOLD_MINIMUM

```
#define QZ_COMP_THRESHOLD_MINIMUM 128
```

6.1.1.10 QZ_COMPRESSED_SZ_OF_EMPTY_FILE

```
#define QZ_COMPRESSED_SZ_OF_EMPTY_FILE 34
```

6.1.1.11 QZ_DATA_ERROR

```
#define QZ_DATA_ERROR (-4)
```

Input data was corrupted

6.1.1.12 QZ_DATA_FORMAT_DEFAULT

```
#define QZ_DATA_FORMAT_DEFAULT QZ_DEFLATE_GZIP_EXT
```

6.1.1.13 QZ_DEFLATE

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

used in gzip header to indicate deflate blocks and in session params

6.1.1.14 QZ_DEFLATE_COMP_LVL_MAXIMUM

```
#define QZ_DEFLATE_COMP_LVL_MAXIMUM (12)
```

6.1.1.15 QZ_DEFLATE_COMP_LVL_MINIMUM

```
#define QZ_DEFLATE_COMP_LVL_MINIMUM (1)
```

6.1.1.16 QZ_DIRECTION_DEFAULT

```
#define QZ_DIRECTION_DEFAULT QZ_DIR_BOTH
```

6.1.1.17 QZ_DUPLICATE

```
#define QZ_DUPLICATE (1)
```

Can not process function again. No failure

6.1.1.18 QZ_FAIL

```
#define QZ_FAIL (-2)
```

Unspecified error

6.1.1.19 QZ_FORCE_SW

```
#define QZ_FORCE_SW (2)
```

Using SW: Switch to software because of previous block

6.1.1.20 QZ_HUFF_HDR_DEFAULT

```
#define QZ_HUFF_HDR_DEFAULT QZ_DYNAMIC_HDR
```

6.1.1.21 QZ_HW_BUFF_MAX_SZ

```
#define QZ_HW_BUFF_MAX_SZ (512*1024)
```

6.1.1.22 QZ_HW_BUFF_MAX_SZ_SPR

```
#define QZ_HW_BUFF_MAX_SZ_SPR (2*1024*1024*1024)
```

6.1.1.23 QZ_HW_BUFF_MIN_SZ

```
#define QZ_HW_BUFF_MIN_SZ (1*1024)
```

6.1.1.24 QZ_HW_BUFF_SZ

```
#define QZ_HW_BUFF_SZ (64*1024)
```

6.1.1.25 QZ_HW_BUFF_SZ_SPR

```
#define QZ_HW_BUFF_SZ_SPR (1*1024*1024)
```

6.1.1.26 QZ_HW_TIMEOUT

```
#define QZ_HW_TIMEOUT(
              ret,
              ext_rc ) (!ret && (ext_rc & QZ_TIMEOUT_MASK))
```

6.1.1.27 QZ_INTEG

```
#define QZ_INTEG (-100)
```

Integrity checked failed

6.1.1.28 QZ_LOW_DEST_MEM

```
#define QZ_LOW_DEST_MEM (15)
```

Using SW: Not enough pinned memory

6.1.1.29 QZ_LOW_MEM

```
#define QZ_LOW_MEM (14)
```

Using SW: Not enough pinned memory

6.1.1.30 QZ_LZ4

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

6.1.1.31 QZ_LZ4s

```
#define QZ_LZ4s ((unsigned char)'s')
```

6.1.1.32 QZ_LZS_COMP_LVL_MAXIMUM

```
#define QZ_LZS_COMP_LVL_MAXIMUM (12)
```

6.1.1.33 QZ_LZS_COMP_LVL_MINIMUM

```
#define QZ_LZS_COMP_LVL_MINIMUM (1)
```

6.1.1.34 QZ_MAX_ALGORITHMS

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

6.1.1.35 QZ_MAX_FORK_DEFAULT

```
#define QZ_MAX_FORK_DEFAULT 3
```

6.1.1.36 QZ_NO_HW

```
#define QZ_NO_HW (11)
```

Using SW: No QAT HW detected

6.1.1.37 QZ_NO_INST_ATTACH

```
#define QZ_NO_INST_ATTACH (13)
```

Using SW: Could not attach to an instance

6.1.1.38 QZ_NO_MDRV

```
#define QZ_NO_MDRV (12)
```

Using SW: No memory driver detected

6.1.1.39 QZ_NO_SW_AVAIL

```
#define QZ_NO_SW_AVAIL (-105)
```

Session may require software, but no software is available

6.1.1.40 QZ_NONE

```
#define QZ_NONE (100)
```

Device uninitialized

6.1.1.41 QZ_NOSW_LOW_MEM

```
#define QZ_NOSW_LOW_MEM (-104)
```

Not using SW: not enough pinned memory

6.1.1.42 QZ_NOSW_NO_HW

```
#define QZ_NOSW_NO_HW (-101)
```

Not using SW: No QAT HW detected

6.1.1.43 QZ_NOSW_NO_INST_ATTACH

```
#define QZ_NOSW_NO_INST_ATTACH (-103)
```

Not using SW: Could not attach to instance

6.1.1.44 QZ_NOSW_NO_MDRV

```
#define QZ_NOSW_NO_MDRV (-102)
```

Not using SW: No memory driver detected

6.1.1.45 QZ_NOSW_UNSUPPORTED_FMT

```
#define QZ_NOSW_UNSUPPORTED_FMT (-116)
```

Not using SW: QAT device does not support data format

6.1.1.46 QZ_PARAMS

```
#define QZ_PARAMS (-1)
```

Invalid parameter in function call

6.1.1.47 QZ_PERIODICAL_POLLING

```
#define QZ_PERIODICAL_POLLING (false)
```

6.1.1.48 QZ_POLL_SLEEP_DEFAULT

```
#define QZ_POLL_SLEEP_DEFAULT 10
```

6.1.1.49 QZ_POST_PROCESS_ERROR

```
#define QZ_POST_PROCESS_ERROR (-117)
```

Using post process: post process callback encountered an error

6.1.1.50 QZ_REQ_THRESHOLD_DEFAULT

```
#define QZ_REQ_THRESHOLD_DEFAULT QZ_REQ_THRESHOLD_MAXIMUM
```

6.1.1.51 QZ_REQ_THRESHOLD_MAXIMUM

```
#define QZ_REQ_THRESHOLD_MAXIMUM NUM_BUFF
```

6.1.1.52 QZ_REQ_THRESHOLD_MINIMUM

```
#define QZ_REQ_THRESHOLD_MINIMUM 1
```

6.1.1.53 QZ_STRM_BUFF_MAX_SZ

```
#define QZ_STRM_BUFF_MAX_SZ (2*1024*1024 - 5*1024)
```

6.1.1.54 QZ_STRM_BUFF_MIN_SZ

```
#define QZ_STRM_BUFF_MIN_SZ (1*1024)
```

6.1.1.55 QZ_STRM_BUFF_SZ_DEFAULT

```
#define QZ_STRM_BUFF_SZ_DEFAULT QZ_HW_BUFF_SZ
```

6.1.1.56 QZ_SW_BACKUP_DEFAULT

```
#define QZ_SW_BACKUP_DEFAULT 1
```

6.1.1.57 QZ_SW_EXECUTION

```
#define QZ_SW_EXECUTION(
              ret,
              ext_rc ) (!ret && (ext_rc & QZ_SW_EXECUTION_MASK))
```

6.1.1.58 QZ_SW_EXECUTION_MASK

```
#define QZ_SW_EXECUTION_MASK (1 << QZ_SW_EXECUTION_BIT)
```

6.1.1.59 QZ_TIMEOUT

```
#define QZ_TIMEOUT (-5)
```

Operation timed out

6.1.1.60 QZ_TIMEOUT_BIT

```
#define QZ_TIMEOUT_BIT (8)
```

6.1.1.61 QZ_TIMEOUT_MASK

```
#define QZ_TIMEOUT_MASK (1 << QZ_TIMEOUT_BIT)
```

6.1.1.62 QZ_UNSUPPORTED_FMT

```
#define QZ_UNSUPPORTED_FMT (16)
```

Using SW: QAT device does not support data format

6.1.1.63 QZ_WAIT_CNT_THRESHOLD_DEFAULT

```
#define QZ_WAIT_CNT_THRESHOLD_DEFAULT 8
```

6.1.1.64 QZ_ZSTD

```
#define QZ_ZSTD ((unsigned char)'Z')
```

6.1.2 Function Documentation

6.1.2.1 qzCompressCrcExt()

```
QATZIP_API int qzCompressCrcExt (
            QzSession_T * sess,
            const unsigned char * src,
            unsigned int * src_len,
            unsigned char * dest,
            unsigned int * dest_len,
            unsigned int last,
            unsigned long * crc,
            uint64_t * ext_rc )
```

6.1.2.2 qzCompressExt()

```
QATZIP_API int qzCompressExt (
            QzSession_T * sess,
            const unsigned char * src,
            unsigned int * src_len,
            unsigned char * dest,
            unsigned int * dest_len,
            unsigned int last,
            uint64_t * ext_rc )
```

6.1.2.3 qzDecompressExt()

```
QATZIP_API int qzDecompressExt (
             QzSession_T * sess,
             const unsigned char * src,
             unsigned int * src_len,
             unsigned char * dest,
             unsigned int * dest_len,
             uint64_t * ext_rc )
```

6.1.2.4 qzMaxCompressedLength()

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

Index