

QATzip
2.0.0

Generated by Doxygen 1.9.8

1 Topic Index	1
1.1 Topics	1
2 Data Structure Index	3
2.1 Data Structures	3
3 File Index	5
3.1 File List	5
4 Topic Documentation	7
4.1 Data Compression API	7
4.1.1 Detailed Description	9
4.1.2 Macro Definition Documentation	9
4.1.2.1 QATZIP_API_VERSION_NUM_MAJOR	9
4.1.2.2 QATZIP_API_VERSION_NUM_MINOR	9
4.1.2.3 QZ_OK	10
4.1.2.4 QZ_SW_BACKUP_BIT_POSITION	10
4.1.2.5 QZ_SW_EXECUTION_BIT	10
4.1.2.6 QZ_MAX_STRING_LENGTH	11
4.1.2.7 QZ_INPUT_CRC_VALID_BIT	11
4.1.2.8 QZ_SKID_PAD_SZ	11
4.1.3 Typedef Documentation	12
4.1.3.1 qzLZ4SCallbackFn	12
4.1.3.2 qzAsyncCallbackFn	14
4.1.3.3 QzMetadataBlob_T	15
4.1.4 Enumeration Type Documentation	15
4.1.4.1 QzHuffmanHdr_T	15
4.1.4.2 PinMem_T	16
4.1.4.3 QzDirection_T	17
4.1.4.4 QzDataFormat_T	17
4.1.4.5 QzPollingMode_T	18
4.1.4.6 QzCrcType_T	18
4.1.4.7 QzSoftwareComponentType_T	18
4.1.4.8 QzLogLevel_T	19
4.1.5 Function Documentation	19
4.1.5.1 qzSetLogLevel()	19
4.1.5.2 qzInit()	21
4.1.5.3 qzSetupSession()	22
4.1.5.4 qzCompress()	24
4.1.5.5 qzCompressCrc()	25
4.1.5.6 qzCompressWithMetadataExt()	27
4.1.5.7 qzCompressWithDictionary()	29
4.1.5.8 qzDecompress()	31

4.1.5.9 qzDecompressCrc()	32
4.1.5.10 qzDecompressWithDictionary()	34
4.1.5.11 qzDecompressWithMetadataExt()	35
4.1.5.12 qzDecompress2()	37
4.1.5.13 qzTeardownSession()	39
4.1.5.14 qzClose()	40
4.1.5.15 qzCompress2()	41
4.1.5.16 qzGetStatus()	43
4.1.5.17 qzSetDefaults()	45
4.1.5.18 qzGetDefaults()	46
4.1.5.19 qzMalloc()	47
4.1.5.20 qzAllocateMetadata()	48
4.1.5.21 qzFree()	50
4.1.5.22 qzFreeMetadata()	51
4.1.5.23 qzMemFindAddr()	52
4.1.5.24 qzCompressStream()	53
4.1.5.25 qzDecompressStream()	55
4.1.5.26 qzEndStream()	56
4.1.5.27 qzGetSoftwareComponentVersionList()	58
4.1.5.28 qzGetSoftwareComponentCount()	59
4.1.5.29 qzGetSessionCrc64Config()	60
4.1.5.30 qzGetSessionCrc32Config()	62
4.1.5.31 qzSetSessionCrc64Config()	63
4.1.5.32 qzSetSessionCrc32Config()	64
4.1.5.33 qzMetadataBlockRead()	66
4.1.5.34 qzMetadataBlockWrite()	67
4.1.5.35 qzMetadataBlockGetCrc64()	69
4.1.5.36 qzMetadataBlockGetCrc32()	70
5 Data Structure Documentation	73
5.1 QzCrc32Config_T Struct Reference	73
5.1.1 Detailed Description	73
5.1.2 Field Documentation	73
5.1.2.1 polynomial	73
5.1.2.2 initial_value	74
5.1.2.3 reflect_in	74
5.1.2.4 reflect_out	74
5.1.2.5 xor_out	74
5.2 QzCrc64Config_T Struct Reference	74
5.2.1 Detailed Description	74
5.2.2 Field Documentation	75
5.2.2.1 polynomial	75

5.2.2.2 initial_value	75
5.2.2.3 reflect_in	75
5.2.2.4 reflect_out	75
5.2.2.5 xor_out	75
5.3 QzCrcResult_T Struct Reference	75
5.3.1 Detailed Description	76
5.3.2 Field Documentation	76
5.3.2.1 status	76
5.3.2.2 valid_flags	76
5.3.2.3 crc_32	76
5.3.2.4 crc_64	76
5.3.2.5 [union]	76
5.3.2.6 [union]	76
5.4 QzDictionaryParams_T Struct Reference	77
5.4.1 Field Documentation	77
5.4.1.1 dictionary	77
5.4.1.2 dictionary_len	77
5.4.1.3 compressed	77
5.4.1.4 crc64Addr	77
5.4.1.5 dictId	77
5.5 QzResult_T Struct Reference	77
5.5.1 Detailed Description	78
5.5.2 Field Documentation	78
5.5.2.1 status	78
5.5.2.2 cb_tag	78
5.5.2.3 src_len	78
5.5.2.4 dest_len	78
5.5.2.5 ext_rc	79
5.5.2.6 crc	79
5.5.2.7 extension_result	79
5.6 QzSession_T Struct Reference	79
5.6.1 Detailed Description	79
5.6.2 Field Documentation	79
5.6.2.1 hw_session_stat	79
5.6.2.2 thd_sess_stat	80
5.6.2.3 internal	80
5.6.2.4 total_in	80
5.6.2.5 total_out	80
5.7 QzSessionParams_T Struct Reference	80
5.7.1 Detailed Description	81
5.7.2 Field Documentation	81
5.7.2.1 huffman_hdr	81

5.7.2.2 direction	81
5.7.2.3 data_fmt	81
5.7.2.4 comp_lvl	81
5.7.2.5 comp_algorithm	81
5.7.2.6 max_forks	81
5.7.2.7 sw_backup	82
5.7.2.8 hw_buff_sz	82
5.7.2.9 strm_buff_sz	82
5.7.2.10 input_sz_thrshold	82
5.7.2.11 req_cnt_thrshold	82
5.7.2.12 wait_cnt_thrshold	82
5.8 QzSessionParamsCommon_T Struct Reference	82
5.8.1 Field Documentation	83
5.8.1.1 direction	83
5.8.1.2 comp_lvl	83
5.8.1.3 comp_algorithm	83
5.8.1.4 max_forks	83
5.8.1.5 sw_backup	83
5.8.1.6 hw_buff_sz	84
5.8.1.7 strm_buff_sz	84
5.8.1.8 input_sz_thrshold	84
5.8.1.9 req_cnt_thrshold	84
5.8.1.10 wait_cnt_thrshold	84
5.8.1.11 polling_mode	84
5.8.1.12 is_sensitive_mode	84
5.9 QzSessionParamsDeflate_T Struct Reference	84
5.9.1 Field Documentation	85
5.9.1.1 common_params	85
5.9.1.2 huffman_hdr	85
5.9.1.3 data_fmt	85
5.10 QzSessionParamsDeflateExt_T Struct Reference	85
5.10.1 Field Documentation	85
5.10.1.1 deflate_params	85
5.10.1.2 stop_decompression_stream_end	85
5.10.1.3 zlib_format	86
5.11 QzSessionParamsLZ4_T Struct Reference	86
5.11.1 Field Documentation	86
5.11.1.1 common_params	86
5.12 QzSessionParamsLZ4S_T Struct Reference	86
5.12.1 Field Documentation	86
5.12.1.1 common_params	86
5.12.1.2 qzCallback	86

5.12.1.3 qzCallback_external	87
5.12.1.4 lz4s_mini_match	87
5.13 QzSessionParamsZstd_T Struct Reference	87
5.13.1 Field Documentation	87
5.13.1.1 common_params	87
5.13.1.2 historyBuffer	87
5.13.1.3 blockSize	87
5.13.1.4 reserved	87
5.14 QzSoftwareVersionInfo_T Struct Reference	88
5.14.1 Field Documentation	88
5.14.1.1 component_type	88
5.14.1.2 component_name	88
5.14.1.3 major_version	88
5.14.1.4 minor_version	88
5.14.1.5 patch_version	88
5.14.1.6 build_number	88
5.14.1.7 reserved	88
5.15 QzStatus_T Struct Reference	89
5.15.1 Detailed Description	89
5.15.2 Field Documentation	89
5.15.2.1 qat_hw_count	89
5.15.2.2 qat_service_init	89
5.15.2.3 qat_mem_drvr	89
5.15.2.4 qat_instance_attach	90
5.15.2.5 memory_allocated	90
5.15.2.6 using_huge_pages	90
5.15.2.7 hw_session_status	90
5.15.2.8 algo_sw	90
5.15.2.9 algo_hw	90
5.16 QzStream_T Struct Reference	90
5.16.1 Detailed Description	91
5.16.2 Field Documentation	91
5.16.2.1 in_sz	91
5.16.2.2 out_sz	91
5.16.2.3 in	91
5.16.2.4 out	91
5.16.2.5 pending_in	92
5.16.2.6 pending_out	92
5.16.2.7 crc_type	92
5.16.2.8 crc_32	92
5.16.2.9 reserved	92
5.16.2.10 opaque	92

6 File Documentation	93
6.1 qatzip.h File Reference	93
6.1.1 Macro Definition Documentation	97
6.1.1.1 QATZIP_API_VERSION	97
6.1.1.2 QATZIP_API	98
6.1.1.3 QZ_DUPLICATE	98
6.1.1.4 QZ_FORCE_SW	98
6.1.1.5 QZ_PARAMS	98
6.1.1.6 QZ_FAIL	98
6.1.1.7 QZ_BUF_ERROR	98
6.1.1.8 QZ_DATA_ERROR	98
6.1.1.9 QZ_TIMEOUT	99
6.1.1.10 QZ_INTEG	99
6.1.1.11 QZ_NO_HW	99
6.1.1.12 QZ_NO_MDRV	99
6.1.1.13 QZ_NO_INST_ATTACH	99
6.1.1.14 QZ_LOW_MEM	99
6.1.1.15 QZ_LOW_DEST_MEM	99
6.1.1.16 QZ_UNSUPPORTED_FMT	99
6.1.1.17 QZ_NONE	100
6.1.1.18 QZ_NOSW_NO_HW	100
6.1.1.19 QZ_NOSW_NO_MDRV	100
6.1.1.20 QZ_NOSW_NO_INST_ATTACH	100
6.1.1.21 QZ_NOSW_LOW_MEM	100
6.1.1.22 QZ_NO_SW_AVAIL	100
6.1.1.23 QZ_NOSW_UNSUPPORTED_FMT	100
6.1.1.24 QZ_POST_PROCESS_ERROR	100
6.1.1.25 QZ_METADATA_OVERFLOW	101
6.1.1.26 QZ_OUT_OF_RANGE	101
6.1.1.27 QZ_NOT_SUPPORTED	101
6.1.1.28 QZ_MAX_ALGORITHMS	101
6.1.1.29 QZ_DEFLATE	101
6.1.1.30 QZ_LZ4	101
6.1.1.31 QZ_LZ4_BLOCK	101
6.1.1.32 QZ_LZ4s	101
6.1.1.33 QZ_ZSTD	101
6.1.1.34 MIN	102
6.1.1.35 QZ_HUFF_HDR_DEFAULT	102
6.1.1.36 QZ_DIRECTION_DEFAULT	102
6.1.1.37 QZ_DATA_FORMAT_DEFAULT	102
6.1.1.38 QZ_COMP_LEVEL_DEFAULT	102
6.1.1.39 QZ_COMP_ALGOL_DEFAULT	102

6.1.1.40 QZ_POLL_SLEEP_DEFAULT	102
6.1.1.41 QZ_MAX_FORK_DEFAULT	102
6.1.1.42 QZ_SW_BACKUP_DEFAULT	102
6.1.1.43 QZ_HW_BUFF_SZ	102
6.1.1.44 QZ_HW_BUFF_SZ_Gen3	103
6.1.1.45 QZ_HW_BUFF_MIN_SZ	103
6.1.1.46 QZ_HW_BUFF_MAX_SZ	103
6.1.1.47 QZ_HW_BUFF_MAX_SZ_Gen3	103
6.1.1.48 QZ_STRM_BUFF_SZ_DEFAULT	103
6.1.1.49 QZ_STRM_BUFF_MIN_SZ	103
6.1.1.50 QZ_STRM_BUFF_MAX_SZ	103
6.1.1.51 QZ_COMP_THRESHOLD_DEFAULT	103
6.1.1.52 QZ_COMP_THRESHOLD_MINIMUM	103
6.1.1.53 QZ_REQ_THRESHOLD_MINIMUM	103
6.1.1.54 QZ_REQ_THRESHOLD_MAXIMUM	104
6.1.1.55 QZ_REQ_THRESHOLD_DEFAULT	104
6.1.1.56 QZ_WAIT_CNT_THRESHOLD_DEFAULT	104
6.1.1.57 QZ_DEFLATE_COMP_LVL_MINIMUM	104
6.1.1.58 QZ_DEFLATE_COMP_LVL_MAXIMUM	104
6.1.1.59 QZ_DEFLATE_COMP_LVL_MAXIMUM_Gen3	104
6.1.1.60 QZ_LZS_COMP_LVL_MINIMUM	104
6.1.1.61 QZ_LZS_COMP_LVL_MAXIMUM	104
6.1.1.62 QZ_AUTO_SELECT_NUMA_NODE	104
6.1.1.63 QZ_ZSTD_COMP_LVL_MINIMUM	104
6.1.1.64 QZ_ZSTD_COMP_LVL_MAXIMUM	105
6.1.1.65 QZ_ZSTD_HIST_BUFF_DEFAULT	105
6.1.1.66 QZ_ZSTD_BLOCKSIZE_DEFAULT	105
6.1.1.67 QZ_ZSTD_HW_BUFF_SZ	105
6.1.1.68 QZ_SW_FORCESW_BIT_POSITION	105
6.1.1.69 QZ_ENABLE_SOFTWARE_BACKUP	105
6.1.1.70 QZ_ENABLE_SOFTWARE_ONLY_EXECUTION	105
6.1.1.71 QZ_DISABLE_SOFTWARE_BACKUP	105
6.1.1.72 QZ_DISABLE_SOFTWARE_ONLY_EXECUTION	105
6.1.1.73 QZ_SW_EXECUTION_MASK	106
6.1.1.74 QZ_SW_EXECUTION	106
6.1.1.75 QZ_TIMEOUT_BIT	106
6.1.1.76 QZ_TIMEOUT_MASK	106
6.1.1.77 QZ_HW_TIMEOUT	106
6.1.1.78 QZ_POST_PROCESS_FAIL_BIT	106
6.1.1.79 QZ_POST_PROCESS_FAIL_MASK	106
6.1.1.80 QZ_POST_PROCESS_FAIL	106
6.1.1.81 QZ_INPUT_CRC_VALID_MASK	106

6.1.1.82 QZ_OUTPUT_CRC_VALID_BIT	107
6.1.1.83 QZ_OUTPUT_CRC_VALID_MASK	107
6.1.1.84 QZ_CRC32_VALID_BIT	107
6.1.1.85 QZ_CRC32_VALID_MASK	107
6.1.1.86 QZ_CRC64_VALID_BIT	107
6.1.1.87 QZ_CRC64_VALID_MASK	107
6.1.1.88 QZ_CRC32_VALID	107
6.1.1.89 QZ_CRC64_VALID	107
6.1.1.90 QZ_INPUT_CRC_VALID	107
6.1.1.91 QZ_OUTPUT_CRC_VALID	107
6.1.1.92 QZ_COMPRESSED_SZ_OF_EMPTY_FILE	108
6.1.2 Function Documentation	108
6.1.2.1 qzSetupSessionDeflate()	108
6.1.2.2 qzSetupSessionLZ4()	108
6.1.2.3 qzSetupSessionLZ4Block()	108
6.1.2.4 qzSetupSessionLZ4S()	108
6.1.2.5 qzSetupSessionZstd()	108
6.1.2.6 qzSetupSessionDeflateExt()	108
6.1.2.7 qzCompressExt()	109
6.1.2.8 qzCompressCrcExt()	109
6.1.2.9 qzCompressCrc64()	109
6.1.2.10 qzCompressCrc64Ext()	109
6.1.2.11 qzDecompressExt()	109
6.1.2.12 qzDecompressCrcExt()	110
6.1.2.13 qzDecompressCrc64()	110
6.1.2.14 qzDecompressCrc64Ext()	110
6.1.2.15 qzGetDeflateEndOfStream()	110
6.1.2.16 qzMaxCompressedLength()	110
6.1.2.17 qzSetDefaultsDeflate()	110
6.1.2.18 qzSetDefaultsLZ4()	111
6.1.2.19 qzSetDefaultsLZ4Block()	111
6.1.2.20 qzSetDefaultsLZ4S()	111
6.1.2.21 qzSetDefaultsZstd()	111
6.1.2.22 qzSetDefaultsDeflateExt()	111
6.1.2.23 qzGetDefaultsDeflate()	111
6.1.2.24 qzGetDefaultsLZ4()	111
6.1.2.25 qzGetDefaultsLZ4Block()	111
6.1.2.26 qzGetDefaultsLZ4S()	111
6.1.2.27 qzGetDefaultsZstd()	112
6.1.2.28 qzGetDefaultsDeflateExt()	112
6.2 qatzip.h	112

Chapter 1

Topic Index

1.1 Topics

Here is a list of all topics with brief descriptions:

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

Chapter 2

Data Structure Index

2.1 Data Structures

Here are the data structures with brief descriptions:

QzCrc32Config_T	73
QzCrc64Config_T	74
QzCrcResult_T	75
QzDictionaryParams_T	77
QzResult_T	77
QzSession_T	79
QzSessionParams_T	80
QzSessionParamsCommon_T	82
QzSessionParamsDeflate_T	84
QzSessionParamsDeflateExt_T	85
QzSessionParamsLZ4_T	86
QzSessionParamsLZ4S_T	86
QzSessionParamsZstd_T	87
QzSoftwareVersionInfo_T	88
QzStatus_T	89
QzStream_T	90

Chapter 3

File Index

3.1 File List

Here is a list of all files with brief descriptions:

qatzip.h	93
------------------------------------	----

Chapter 4

Topic Documentation

4.1 Data Compression API

Data Structures

- struct [QzSessionParams_T](#)
- struct [QzSession_T](#)
- struct [QzStatus_T](#)
- struct [QzCrc64Config_T](#)
- struct [QzCrc32Config_T](#)
- struct [QzCrcResult_T](#)
- struct [QzResult_T](#)
- struct [QzStream_T](#)

Macros

- #define [QATZIP_API_VERSION_NUM_MAJOR](#) (2)
- #define [QATZIP_API_VERSION_NUM_MINOR](#) (6)
- #define [QZ_OK](#) (0)
- #define [QZ_SW_BACKUP_BIT_POSITION](#) (0)
- #define [QZ_SW_EXECUTION_BIT](#) (4)
- #define [QZ_MAX_STRING_LENGTH](#) 64
- #define [QZ_INPUT_CRC_VALID_BIT](#) (4)
- #define [QZ_SKID_PAD_SZ](#) 48

Typedefs

- typedef int(* [qzLZ4SCallbackFn](#)) (void *external, const unsigned char *src, unsigned int *src_len, unsigned char *dest, unsigned int *dest_len, int *ExtStatus)
- typedef int(* [qzAsyncCallbackFn](#)) ([QzResult_T](#) *res)
- typedef void * [QzMetadataBlob_T](#)

Enumerations

- enum `QzHuffmanHdr_T` { `QZ_DYNAMIC_HDR` = 0 , `QZ_STATIC_HDR` }
- enum `PinMem_T` { `COMMON_MEM` = 0 , `PINNED_MEM` }
- enum `QzDirection_T` { `QZ_DIR_COMPRESS` = 0 , `QZ_DIR_DECOMPRESS` , `QZ_DIR_BOTH` }
- enum `QzDataFormat_T` {
 `QZ_DEFLATE_4B` = 0 , `QZ_DEFLATE_GZIP` , `QZ_DEFLATE_GZIP_EXT` , `QZ_DEFLATE_RAW` ,
 `QZ_FMT_NUM` }
- enum `QzPollingMode_T` { `QZ_PERIODICAL_POLLING` = 0 , `QZ_BUSY_POLLING` }
- enum `QzCrcType_T` { `QZ_CRC32` = 0 , `QZ_ADLER` , `QZ_XXHASH` , `NONE` }
- enum `QzSoftwareComponentType_T` {
 `QZ_COMPONENT_FIRMWARE` = 0 , `QZ_COMPONENT_KERNEL_DRIVER` , `QZ_COMPONENT_USER_DRIVER`
 , `QZ_COMPONENT_QATZIP_API` ,
 `QZ_COMPONENT_SOFTWARE_PROVIDER` }
- enum `QzLogLevel_T` {
 `LOG_NONE` = 0 , `LOG_FATAL` , `LOG_ERROR` , `LOG_WARNING` ,
 `LOG_INFO` , `LOG_DEBUG1` , `LOG_DEBUG2` , `LOG_DEBUG3` }

Functions

- `QzLogLevel_T qzSetLogLevel (QzLogLevel_T level)`
- `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 qzCompressWithMetadataExt (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, QzMetadataBlob_T metadata, uint32_t hw_buff_sz_override, uint32_t comp_thrsld)`
- `int qzCompressWithDictionary (QzSession_T *sess, const unsigned char *src, unsigned char *dest, QzDictionaryParams_T *dictionary, qzAsyncCallbackFn callback, QzResult_T *qzResults)`
- `int qzDecompress (QzSession_T *sess, const unsigned char *src, unsigned int *src_len, unsigned char *dest, unsigned int *dest_len)`
- `int qzDecompressCrc (QzSession_T *sess, const unsigned char *src, unsigned int *src_len, unsigned char *dest, unsigned int *dest_len, unsigned long *crc)`
- `int qzDecompressWithDictionary (QzSession_T *sess, const unsigned char *src, unsigned char *dest, QzDictionaryParams_T *dictionary, qzAsyncCallbackFn callback, QzResult_T *qzResults)`
- `int qzDecompressWithMetadataExt (QzSession_T *sess, const unsigned char *src, unsigned int *src_len, unsigned char *dest, unsigned int *dest_len, uint64_t *ext_rc, QzMetadataBlob_T metadata, uint32_t hw_buff_sz_override)`
- `int qzDecompress2 (QzSession_T *sess, const unsigned char *src, unsigned char *dest, qzAsyncCallbackFn callback, QzResult_T *qzResults)`
- `int qzTeardownSession (QzSession_T *sess)`
- `int qzClose (QzSession_T *sess)`
- `int qzCompress2 (QzSession_T *sess, const unsigned char *src, unsigned char *dest, qzAsyncCallbackFn callback, QzResult_T *qzResults)`
- `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)`
- `int qzAllocateMetadata (QzMetadataBlob_T *metadata, size_t data_size, uint32_t hw_buff_sz)`
- `void qzFree (void *m)`
- `int qzFreeMetadata (QzMetadataBlob_T metadata)`
- `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)`
- `int qzGetSoftwareComponentVersionList (QzSoftwareVersionInfo_T *api_info, unsigned int *num_elem)`
- `int qzGetSoftwareComponentCount (unsigned int *num_elem)`
- `int qzGetSessionCrc64Config (QzSession_T *sess, QzCrc64Config_T *crc64_config)`
- `int qzGetSessionCrc32Config (QzSession_T *sess, QzCrc32Config_T *crc32_config)`
- `int qzSetSessionCrc64Config (QzSession_T *sess, QzCrc64Config_T *crc64_config)`
- `int qzSetSessionCrc32Config (QzSession_T *sess, QzCrc32Config_T *crc32_config)`
- `int qzMetadataBlockRead (uint32_t block_num, QzMetadataBlob_T metadata, uint32_t *block_offset, uint32_t *block_size, uint32_t *block_flags, uint32_t *block_hash)`
- `int qzMetadataBlockWrite (uint32_t block_num, QzMetadataBlob_T metadata, uint32_t *block_offset, uint32_t *block_size, uint32_t *block_flags, uint32_t *block_hash)`
- `int qzMetadataBlockGetCrc64 (uint32_t block_num, QzMetadataBlob_T metadata, uint64_t *input_crc, uint64_t *output_crc)`
- `int qzMetadataBlockGetCrc32 (uint32_t block_num, QzMetadataBlob_T metadata, uint32_t *input_crc, uint32_t *output_crc)`

4.1.1 Detailed Description

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

Description:

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 (6)
```

QATzip Minor Version Number

Description:

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

Description:

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

4.1.2.4 QZ_SW_BACKUP_BIT_POSITION

```
#define QZ_SW_BACKUP_BIT_POSITION (0)
```

QATzip Session software configuration settings

Description:

The following definitions can be used with the `sw_backup` variable in structs and functions to configure the session

<code>QZ_ENABLE_SOFTWARE_BACKUP</code>	Configure session with software fallback
<code>QZ_ENABLE_SOFTWARE_ONLY_EXECUTION</code>	Configure session to only use software

4.1.2.5 QZ_SW_EXECUTION_BIT

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

< Disable SW only compression/decompression operations

QATzip Extended return information

Description:

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.

If set in the extended return value, `QZ_POST_PROCESS_FAIL` indicates post processing of the LZ4s compressed data has failed.

4.1.2.6 QZ_MAX_STRING_LENGTH

```
#define QZ_MAX_STRING_LENGTH 64
```

QATzip software version structure

Description:

This structure contains data relating to the versions of a QATZip or a subcomponent of this library platform.

4.1.2.7 QZ_INPUT_CRC_VALID_BIT

```
#define QZ_INPUT_CRC_VALID_BIT (4)
```

QATzip crc structure's valid_flags bitmap

Description:

The following definitions can be used with the crc structure's valid_flags.

QZ_INPUT_CRC_VALID_MASK indicates if input crc feild is valid
QZ_OUTPUT_CRC_VALID_MASK indicates if output crc feild is valid
QZ_CRC32_VALID_MASK indicates if crc type is crc32 in union
QZ_CRC64_VALID_MASK indicates if crc type is crc64 in union

4.1.2.8 QZ_SKID_PAD_SZ

```
#define QZ_SKID_PAD_SZ 48
```

Get the maximum compressed output length

Description:

Get the maximum compressed output length.

Context:

This function shall not be called in an interrupt context.

Assumptions:

None

Side Effects:

None

Blocking:

Yes

Reentrant:

No

Thread Safe:

Yes

Parameters

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

Return values

<code>dest_sz</code>	Max compressed data output length in bytes. When <code>src_sz</code> 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

4.1.3 Typedef Documentation**4.1.3.1 qzLZ4SCallbackFn**

```
typedef int(* qzLZ4SCallbackFn) (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**Description:**

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 `qzLZ4SCallbackFn` return 'QZ_POST_PROCESS_ERROR'. See extended return code section and *Ext API for details.

Context:

This function shall not be called in an interrupt context.

Assumptions:

None

Side Effects:

None

Blocking:

Yes

Reentrant:

No

Thread Safe:

Yes

Parameters

in	<i>external</i>	User context provided through the 'qzCallback_external' pointer in the QzSessionParams_T structure.
in	<i>src</i>	Point to source buffer
in, out	<i>src_len</i>	Length of source buffer. Modified to number of bytes consumed
in	<i>dest</i>	Point to destination buffer
in, out	<i>dest_len</i>	Length of destination buffer. Modified to length of compressed data when function returns
in, out	<i>ExtStatus</i>	'qzCallback' customized error code.

Return values

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

Precondition

None

Postcondition

None

Note

Only a synchronous version of this function is provided.

See also

None

4.1.3.2 qzAsyncCallbackFn

```
typedef int(* qzAsyncCallbackFn) (QzResult_T *res)
```

Description:

This callback function will be called in asynchronous compression and decompression API. Function implementation should be provided by user and comply with this prototype's rules.

Context:

This function shall not be called in an interrupt context.

Assumptions:

None

Side Effects:

None

Blocking:

Yes

Reentrant:

No

Thread Safe:

Yes

Parameters

<i>in, out</i>	<i>res</i>	it's same as compress2/decompress2 API's last parameters, include offloading result.
----------------	------------	--

Precondition

None

Postcondition

None

Note

None

See also

None

4.1.3.3 QzMetadataBlob_T

```
typedef void* QzMetadataBlob_T
```

QATzip pointer to opaque metadata.

Description:

The opaque pointer to metadata.

4.1.4 Enumeration Type Documentation**4.1.4.1 QzHuffmanHdr_T**

```
enum 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, NULL); qzTeardownSession(&sess); qzClose(&sess);

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

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

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

If an application terminates without invoking tear down and close functions, process termination will invoke memory and hardware instance cleanup.

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

Changes from Gen4 onwards

1. Addition of LZ4 and LZ4s
2. Addition of post-processing functions for LZ4s
3. Compression level up to 12 for LZ4 and LZ4s
4. Support for gzip header with additional compression algorithms

Changes from Gen6 onwards

1. Addition of ZSTD for both comp and decomp, no post-processing required.
 2. Compression level up to 12 for ZSTD
 3. Static Huffman is no longer supported. QATzip will fall back to software compression, which may fail if software backup is disabled.
- Supported Huffman Headers

Description:

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

4.1.4.2 PinMem_T

```
enum PinMem_T
```

Supported memory types

Description:

This enumerated list identifies memory types supported by QATzip.

Enumerator

COMMON_MEM	Allocate non-contiguous memory
PINNED_MEM	Allocate contiguous memory

4.1.4.3 QzDirection_T

```
enum QzDirection_T
```

Compress or decompress setting

Description:

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

4.1.4.4 QzDataFormat_T

```
enum QzDataFormat_T
```

Streaming API input and output format

Description:

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_FMT_NUM	

4.1.4.5 QzPollingMode_T

enum [QzPollingMode_T](#)

Supported polling mode

Description:

Specifies whether the instance must be busy polling,
or be periodical polling.

Enumerator

QZ_PERIODICAL_POLLING	No busy polling
QZ_BUSY_POLLING	busy polling

4.1.4.6 QzCrcType_T

enum [QzCrcType_T](#)

Supported checksum type

Description:

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

Enumerator

QZ_CRC32	CRC32 checksum
QZ_ADLER	Adler checksum
QZ_XXHASH	xxhash checksum
NONE	No checksum

4.1.4.7 QzSoftwareComponentType_T

enum [QzSoftwareComponentType_T](#)

Software Component type

Description:

This enumerated list specifies the type of software that is being
described.

Enumerator

QZ_COMPONENT_FIRMWARE	
QZ_COMPONENT_KERNEL_DRIVER	
QZ_COMPONENT_USER_DRIVER	
QZ_COMPONENT_QATZIP_API	
QZ_COMPONENT_SOFTWARE_PROVIDER	

4.1.4.8 QzLogLevel_T

```
enum QzLogLevel_T
```

QATzip log verbosity level

Description:

```
This is an emum for qatzip log verbosity level
```

Enumerator

LOG_NONE	
LOG_FATAL	
LOG_ERROR	
LOG_WARNING	
LOG_INFO	
LOG_DEBUG1	
LOG_DEBUG2	
LOG_DEBUG3	

4.1.5 Function Documentation

4.1.5.1 qzSetLogLevel()

```
QzLogLevel_T qzSetLogLevel (
    QzLogLevel_T level )
```

Set qatzip log verbosity level

Description:

```
Set qatzip log verbosity level
```

Context:

```
This function shall not be called in an interrupt context.
```

Assumptions:

None

Side Effects:

None

Blocking:

Yes

Reentrant:

No

Thread Safe:

Yes

Parameters

<i>in</i>	<i>level</i>	set log verbosity level
-----------	--------------	-------------------------

Return values

<i>Old</i>	log verbosity level
------------	---------------------

Precondition

None

Postcondition

None

Note

None

See also

None

4.1.5.2 qzInit()

```
int qzInit (
    QzSession_T * sess,
    unsigned char sw_backup )
```

Initialize QAT hardware

Description:

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.

Context:

This function shall not be called in an interrupt context.

Assumptions:

None

Side Effects:

This function will:

- 1) start the user space driver if necessary
- 2) allocate all hardware instances available

Blocking:

Yes

Reentrant:

No

Thread Safe:

Yes

Parameters

in	<code>sess</code>	Session handle (pointer to opaque instance and session data.)
in	<code>sw_backup</code>	see <code>QZ_SW_*</code> definitions for expected behavior

Return values

<i>QZ_OK</i>	Function executed successfully. A hardware or software instance has been allocated to the calling process/thread
<i>QZ_DUPLICATE</i>	This process/thread already has a hardware instance
<i>QZ_PARAMS</i>	*sess is NULL
<i>QZ_NOSW_NO_HW</i>	No hardware and no software session being established
<i>QZ_NOSW_NO_MDRV</i>	No memory driver. No software session established
<i>QZ_NOSW_NO_INST_ATTACH</i>	No instance available No software session established
<i>QZ_NOSW_LOW_MEM</i>	Not enough pinned memory available No software session established
<i>QZ_UNSUPPORTED_FMT</i>	No support for requested algorithm; using software
<i>QZ_NOSW_UNSUPPORTED_FMT</i>	No support for requested algorithm; No software session established
<i>QZ_NO_SW_AVAIL</i>	No software is available. This will be returned when sw_backup is set but the session does not support software operations or software fallback is unavailable to the application.

Precondition

None

Postcondition

None

Note

Only a synchronous version of this function is provided.

See also

None

4.1.5.3 qzSetupSession()

```
int qzSetupSession (
    QzSession_T * sess,
    QzSessionParams_T * params )
```

Initialize a QATzip session

Description:

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.

Context:

This function shall not be called in an interrupt context.

Assumptions:

None

Side Effects:

None

Blocking:

Yes

Reentrant:

No

Thread Safe:

Yes

Parameters

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

Return values

<i>QZ_OK</i>	Function executed successfully. A hardware or software based compression session has been created
<i>QZ_DUPLICATE</i>	*sess includes an existing hardware or software session
<i>QZ_PARAMS</i>	*sess is NULL or member of params is invalid
<i>QZ_NOSW_NO_HW</i>	No hardware and no sw session being established
<i>QZ_NOSW_NO_MDRV</i>	No memory driver. No software session established
<i>QZ_NOSW_NO_INST_ATTACH</i>	No instance available No software session established
<i>QZ_NO_LOW_MEM</i>	Not enough pinned memory available No software session established
<i>QZ_UNSUPPORTED_FMT</i>	No support for requested algorithm; using software
<i>QZ_NOSW_UNSUPPORTED_FMT</i>	No support for requested algorithm; No software session established
<i>QZ_NO_SW_AVAIL</i>	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

4.1.5.4 qzCompress()

```
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**Description:**

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.

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.

Context:

This function shall not be called in an interrupt context.

Assumptions:

None

Side Effects:

None

Blocking:

Yes

Reentrant:

No

Thread Safe:

Yes

Parameters

in	<i>sess</i>	Session handle (pointer to opaque instance and session data)
in	<i>src</i>	Point to source buffer
in, out	<i>src_len</i>	Length of source buffer. Modified to number of bytes consumed
in	<i>dest</i>	Point to destination buffer
in, out	<i>dest_len</i>	Length of destination buffer. Modified to length of compressed data when function returns
in	<i>last</i>	1 for 'No more data to be compressed' 0 for 'More data to be compressed'
in, out	<i>ext_rc</i>	qzCompressExt only. If not NULL, <i>ext_rc</i> 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

<i>QZ_OK</i>	Function executed successfully
<i>QZ_FAIL</i>	Function did not succeed
<i>QZ_PARAMS</i>	* <i>sess</i> 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 qzCompressCrc()

```
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

Description:

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*.

This function will place completed compression blocks in the output buffer and put CRC32 or CRC64 checksum for compressed input data in the 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.

Context:

This function shall not be called in an interrupt context.

Assumptions:

None

Side Effects:

None

Blocking:

Yes

Reentrant:

No

Thread Safe:

Yes

Parameters

in	<i>sess</i>	Session handle (pointer to opaque instance and session data)
in	<i>src</i>	Point to source buffer
in, out	<i>src_len</i>	Length of source buffer. Modified to number of bytes consumed
in	<i>dest</i>	Point to destination buffer
in, out	<i>dest_len</i>	Length of destination buffer. Modified to length of compressed data when function returns
in	<i>last</i>	1 for 'No more data to be compressed' 0 for 'More data to be compressed'
in, out	<i>crc</i>	Pointer to CRC32 or CRC64 checksum buffer
in, out	<i>ext_rc</i>	<i>qzCompressCrcExt</i> or <i>qzCompressCrc64Ext</i> only. If not NULL, <i>ext_rc</i> 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

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

Precondition

None

Postcondition

None

Note

Only a synchronous version of this function is provided.

See also

None

4.1.5.6 qzCompressWithMetadataExt()

```
int qzCompressWithMetadataExt (
    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,
    QzMetadataBlob_T metadata,
    uint32_t hw_buff_sz_override,
    uint32_t comp_thrshold )
```

Compress a buffer and write metadata for each compressed block into the opaque metadata structure.

Description:

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.

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.

The metadata for each compressed block will be written into the opaque metadata structure specified as function param `metadata`.

`comp_thrshold` specifies compression threshold of a block. If compressed size of the block is $> \text{comp_thrshold}$, the compression function shall copy the uncompressed data to the output buffer and set the size of the block in the metadata to the size of the uncompressed block. If the compressed size of the block is $\leq \text{comp_thrshold}$, the compressed data will be copied to the output buffer and the compressed size will be set in the metadata.

`hw_buff_sz_override` specifies the data size to be used for the each compression operation. It overrides the `hw_buff_sz` parameter specified at session creation. If 0 is provided for this parameter, then the `hw_buff_sz` specified at session creation will be used. Memory for the opaque metadata structure should be allocated based on the `hw_buff_sz` or the `hw_buff_sz_override` that is used for the compression operation.

Context:

This function shall not be called in an interrupt context.

Assumptions:

None

Side Effects:

None

Blocking:

Yes

Reentrant:

No

Thread Safe:

Yes

Parameters

in	<i>sess</i>	Session handle (pointer to opaque instance and session data)
in	<i>src</i>	Point to source buffer.
in, out	<i>src_len</i>	Length of source buffer. Modified to number of bytes consumed.
in	<i>dest</i>	Point to destination buffer.
in, out	<i>dest_len</i>	Length of destination buffer. Modified to length of compressed data when function returns.
in	<i>last</i>	1 for 'No more data to be compressed' 0 for 'More data to be compressed'
in, out	<i>ext_rc</i>	If not NULL, <i>ext_rc</i> 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.
		Generated by Doxygen
in, out	<i>metadata</i>	Pointer to opaque metadata.
in	<i>hw_buff_sz_override</i>	Data size to be used for compression.
in	<i>comp_thrshold</i>	Compressed block threshold.

Return values

<i>QZ_OK</i>	Function executed successfully
<i>QZ_FAIL</i>	Function did not succeed
<i>QZ_PARAMS</i>	*sess or metadata is NULL or Member of params is invalid, hw_buff_sz_override is invalid data size.
<i>QZ_METADATA_OVERFLOW</i>	Unable to populate metadata due to insufficient memory allocated.
<i>QZ_NOT_SUPPORTED</i>	Compression with metadata is not supported with given algorithm or format.
<i>QZ_NOSW_NO_HW</i>	Function did not find an installed kernel driver or software provider.
<i>QZ_NOSW_NO_INST_ATTACH</i>	No instance available.

Precondition

None

Postcondition

None

Note

Only a synchronous version of this function is provided.

See also

None

4.1.5.7 qzCompressWithDictionary()

```
int qzCompressWithDictionary (
    QzSession_T * sess,
    const unsigned char * src,
    unsigned char * dest,
    QzDictionaryParams_T * dictionary,
    qzAsyncCallbackFn callback,
    QzResult_T * qzResults )
```

Compress a buffer with dictionary (asynchronous or synchronous model)

Description:

These functions has the same compression ability as "qzCompress", but with asynchronous or synchronous model.

If user provide the callback function(callback is not NULL), it would work as asynchronous model. Otherwise, it would work as synchronous model, just like "qzCompress".

Context:

This function shall not be called in an interrupt context.

Assumptions:

None

Side Effects:

None

Blocking:

No

Reentrant:

No

Thread Safe:

Yes

Parameters

in	<i>sess</i>	Session handle (pointer to opaque instance and session data)
in	<i>src</i>	Pointer to source buffer.
in	<i>dest</i>	Pointer to destination buffer.
in	<i>dictionary</i>	Pointer to structure containing dictionary parameters
in	<i>callback</i>	User-defined callback Function to be called upon completion of the compression operation. it could be NULL, then it's an synchronous call, otherwise, it's asynchronous call.
in, out	<i>qzResults</i>	Pointer to QzResult, It have to allocate by user.

Return values

<i>QZ_OK</i>	Request submit successfully.
<i>QZ_FAIL</i>	Request submit failed.
<i>QZ_PARAMS</i>	*sess is NULL or member of params is invalid.

Precondition

None

Postcondition

None

Note

None

See also

None

4.1.5.8 qzDecompress()

```
int qzDecompress (
    QzSession_T * sess,
    const unsigned char * src,
    unsigned int * src_len,
    unsigned char * dest,
    unsigned int * dest_len )
```

Decompress a buffer

Description:

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.

Context:

This function shall not be called in an interrupt context.

Assumptions:

None

Side Effects:

None

Blocking:

Yes

Reentrant:

No

Thread Safe:

Yes

Parameters

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

Return values

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

Precondition

None

Postcondition

None

Note

Only a synchronous version of this function is provided.

See also

None

4.1.5.9 qzDecompressCrc()

```
int qzDecompressCrc (
    QzSession_T * sess,
    const unsigned char * src,
    unsigned int * src_len,
    unsigned char * dest,
    unsigned int * dest_len,
    unsigned long * crc )
```

Decompress a buffer and return the CRC checksum

Description:

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.

This function will place completed decompression chunks in the output buffer and put the CRC32 or CRC64 checksum for compressed input data in the user provided buffer *crc.

Context:

This function shall not be called in an interrupt context.

Assumptions:

None

Side Effects:

None

Blocking:

Yes

Reentrant:

No

Thread Safe:

Yes

Parameters

in	<i>sess</i>	Session handle (pointer to opaque instance and session data)
in	<i>src</i>	Point to source buffer
in	<i>src_len</i>	Length of source buffer. Modified to length of processed compressed data when function returns
in	<i>dest</i>	Point to destination buffer
in, out	<i>dest_len</i>	Length of destination buffer. Modified to length of decompressed data when function returns
in, out	<i>crc</i>	Pointer to CRC32 or CRC64 checksum buffer
in, out	<i>ext_rc</i>	qzDecompressCrcExt or qzDecompressCrc64Ext 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

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

Precondition

None

Postcondition

None

Note

Only a synchronous version of this function is provided.

See also

None

4.1.5.10 qzDecompressWithDictionary()

```
int qzDecompressWithDictionary (
    QzSession_T * sess,
    const unsigned char * src,
    unsigned char * dest,
    QzDictionaryParams_T * dictionary,
    qzAsyncCallbackFn callback,
    QzResult_T * qzResults )
```

Decompress a buffer with dictionary (asynchronous or synchronous model)

Description:

These functions has the same decompression ability as "qzDecompress", but with asynchronous or synchronous model.

NOTE: asynchronous model currently not supported. Supplying a non-null callback function will result in error QZ_NOT_SUPPORTED.

Context:

This function shall not be called in an interrupt context.

Assumptions:

None

Side Effects:

None

Blocking:

No

Reentrant:

No

Thread Safe:

Yes

Parameters

in	<i>sess</i>	Session handle (pointer to opaque instance and session data)
in	<i>src</i>	Pointer to source buffer containing compressed data.
in	<i>dest</i>	Pointer to destination buffer where decompressed data will be stored.
in	<i>dictionary</i>	Pointer to structure containing dictionary parameters
in	<i>callback</i>	User-defined callback function to be called upon completion of the decompression operation.
in, out	<i>qzResults</i>	Pointer to QzResult, It have to allocate by user.

Return values

<i>QZ_OK</i>	Decompression request submitted successfully.
<i>QZ_FAIL</i>	Decompression request submission failed.
<i>QZ_PARAMS</i>	*sess is NULL or member of params is invalid.

Precondition

None

Postcondition

None

Note

None

See also

None

4.1.5.11 qzDecompressWithMetadataExt()

```
int qzDecompressWithMetadataExt (
    QzSession_T * sess,
    const unsigned char * src,
    unsigned int * src_len,
    unsigned char * dest,
    unsigned int * dest_len,
    uint64_t * ext_rc,
    QzMetadataBlob_T metadata,
    uint32_t hw_buff_sz_override )
```

Decompress a buffer with metadata.

Description:

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 content of **sess* - then this function will attempt to set up a session using *qzInit* and *qzSetupSession*.

The metadata function parameter specifies metadata of compressed file which can be used for regular or parallel decompression.

hw_buff_sz_override specifies the data size to be used for the each decompression operation. It overrides the *hw_buff_sz* parameter specified at session creation. If 0 is provided for this parameter, then the *hw_buff_sz* specified at session creation will be used. Memory for the opaque metadata structure should be allocated based on the *hw_buff_sz* or the *hw_buff_sz_override* that is used for the compression operation.

Context:

This function shall not be called in an interrupt context.

Assumptions:

None

Side Effects:

None

Blocking:

Yes

Reentrant:

No

Thread Safe:

Yes

Parameters

in	<i>sess</i>	Session handle (pointer to opaque instance and session data)
in	<i>src</i>	Point to source buffer
in	<i>src_len</i>	Length of source buffer. Modified to length of processed compressed data when function returns
in	<i>dest</i>	Point to destination buffer
in, out	<i>dest_len</i>	Length of destination buffer. Modified to length of decompressed data when function returns
in, out	<i>ext_rc</i>	If not NULL, <i>ext_rc</i> points 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.
in	<i>metadata</i>	Pointer to opaque metadata.
in	<i>hw_buff_sz_override</i>	Expected size of decompressed block.

Return values

<i>QZ_OK</i>	Function executed successfully.
<i>QZ_FAIL</i>	Function did not succeed.
<i>QZ_PARAMS</i>	*sess or metadata is NULL or Member of params is invalid, hw_buff_sz_override is invalid data size.
<i>QZ_METADATA_OVERFLOW</i>	Unable to populate metadata due to insufficient memory allocated.
<i>QZ_NOT_SUPPORTED</i>	Decompression with metadata is not supported with given algorithm or format.
<i>QZ_NOSW_NO_HW</i>	Function did not find an installed kernel driver or software provider.
<i>QZ_NOSW_NO_INST_ATTACH</i>	No instance available.

Precondition

None

Postcondition

None

Note

Only a synchronous version of this function is provided.

See also

None

4.1.5.12 qzDecompress2()

```
int qzDecompress2 (  
    QzSession_T * sess,  
    const unsigned char * src,  
    unsigned char * dest,  
    qzAsyncCallbackFn callback,  
    QzResult_T * qzResults )
```

Decompress a buffer with asynchronous or synchronous model

Description:

These functions has the same decompression ability as "qzDecompress", but with asynchronous or synchronous model.

If user provide the callback function(callback is not NULL), it would work as asynchronous model. Otherwise, it would work as synchronous model, just like "qzDecompress".

Context:

This function shall not be called in an interrupt context.

Assumptions:

None

Side Effects:

None

Blocking:

No

Reentrant:

No

Thread Safe:

Yes

Parameters

in	<i>sess</i>	Session handle (pointer to opaque instance and session data)
in	<i>src</i>	Pointer to source buffer containing compressed data.
in	<i>dest</i>	Pointer to destination buffer where decompressed data will be stored.
in	<i>callback</i>	User-defined callback function to be called upon completion of the decompression operation. it could be NULL, then it's an synchronous call, otherwise, it's asynchronous call.
in, out	<i>qzResults</i>	Pointer to QzResult, It have to allocate by user.

Return values

<i>QZ_OK</i>	Decompression request submitted successfully.
<i>QZ_FAIL</i>	Decompression request submission failed.
<i>QZ_PARAMS</i>	*sess is NULL or member of params is invalid.

Precondition

None

Postcondition

None

Note

None

See also

None

4.1.5.13 qzTeardownSession()

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

Uninitialize a QATzip session

Description:

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

Context:

This function shall not be called in an interrupt context.

Assumptions:

None

Side Effects:

None

Blocking:

Yes

Reentrant:

No

Thread Safe:

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

4.1.5.14 qzClose()

```
int qzClose (
    QzSession_T * sess )
```

Terminates a QATzip session

Description:

This function closes the connection with QAT.

Context:

This function shall not be called in an interrupt context.

Assumptions:

None

Side Effects:

None

Blocking:

Yes

Reentrant:

No

Thread Safe:

Yes

Parameters

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

Return values

<code>QZ_OK</code>	Function executed successfully
<code>QZ_FAIL</code>	Function did not succeed
<code>QZ_PARAMS</code>	*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

4.1.5.15 qzCompress2()

```
int qzCompress2 (
    QzSession_T * sess,
    const unsigned char * src,
    unsigned char * dest,
    qzAsyncCallbackFn callback,
    QzResult_T * qzResults )
```

Compress a buffer with asynchronous or synchronous model

Description:

These functions has the same compression ability as "qzCompress", but with asynchronous or synchronous model.

If user provide the callback function(callback is not NULL), it would work as asynchronous model. Otherwise, it would work as synchronous model, just like "qzCompress".

Context:

This function shall not be called in an interrupt context.

Assumptions:

None

Side Effects:

None

Blocking:

No

Reentrant:

No

Thread Safe:

Yes

Parameters

in	<i>sess</i>	Session handle (pointer to opaque instance and session data)
in	<i>src</i>	Pointer to source buffer.
in	<i>dest</i>	Pointer to destination buffer.
in	<i>callback</i>	User-defined callback Function to be called upon completion of the compression operation. it could be NULL, then it's an synchronous call, otherwise, it's asynchronous call.
in, out	<i>qzResults</i>	Pointer to QzResult, It have to allocate by user.

Return values

<i>QZ_OK</i>	Request submit successfully.
<i>QZ_FAIL</i>	Request submit failed.
<i>QZ_PARAMS</i>	*sess is NULL or member of params is invalid.

Precondition

None

Postcondition

None

Note

None

See also

None

4.1.5.16 qzGetStatus()

```
int qzGetStatus (
    QzSession_T * sess,
    QzStatus_T * status )
```

Get current QAT status

Description:

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_allocated Amount of memory, in kilobytes,
from kernel or huge pages allocated by this process/thread. using_huge_pages 1 if memory is being allocated
from huge pages, 0 if memory is being allocated from standard kernel memory hw_session_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_NOSW_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 be available. If the application relies on software backup being available, then this return code can be treated as an error.

Context:

This function shall not be called in an interrupt context.

Assumptions:

None

Side Effects:

None

Blocking:

Yes

Reentrant:

No

Thread Safe:

Yes

Parameters

in	<i>sess</i>	Session handle (pointer to opaque instance and session data)
in	<i>status</i>	Pointer to QATzip status structure

Return values

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

Precondition

None

Postcondition

None

Note

Only a synchronous version of this function is provided.

See also

None

4.1.5.17 qzSetDefaults()

```
int qzSetDefaults (
    QzSessionParams_T * defaults )
```

Set default `QzSessionParams_T` value

Description:

Set default `QzSessionParams_T` value.

Context:

This function shall not be called in an interrupt context.

Assumptions:

None

Side Effects:

None

Blocking:

Yes

Reentrant:

No

Thread Safe:

Yes

Parameters

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

Return values

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

Precondition

None

Postcondition

None

Note

Only a synchronous version of this function is provided.

See also

None

4.1.5.18 qzGetDefaults()

```
int qzGetDefaults (
    QzSessionParams_T * defaults )
```

Get default [QzSessionParams_T](#) value

Description:

Get default [QzSessionParams_T](#) value.

Context:

This function shall not be called in an interrupt context.

Assumptions:

None

Side Effects:

None

Blocking:

Yes

Reentrant:

No

Thread Safe:

Yes

Parameters

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

Return values

<code>QZ_OK</code>	Success on getting default value
<code>QZ_PARAM</code>	Fail to get default value

Precondition

None

Postcondition

None

Note

Only a synchronous version of this function is provided.

See also

None

4.1.5.19 qzMalloc()

```
void * qzMalloc (
    size_t sz,
    int numa,
    int force_pinned )
```

Allocate different types of memory

Description:

Allocate different types of memory.

Context:

This function shall not be called in an interrupt context.

Assumptions:

None

Side Effects:

None

Blocking:

Yes

Reentrant:

No

Thread Safe:

Yes

Parameters

in	<i>sz</i>	Memory size to be allocated
in	<i>numa</i>	NUMA node (0 or greater) from which to allocate memory QZ_AUTO_SELECT_NUMA_NODE (-1) to auto select optimal NUMA node
in	<i>force_pinned</i>	PINNED_MEM allocate contiguous memory COMMON_MEM allocate non-contiguous memory

Return values

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

Precondition

None

Postcondition

None

Note

Only a synchronous version of this function is provided.

See also

None

4.1.5.20 qzAllocateMetadata()

```
int qzAllocateMetadata (
    QzMetadataBlob_T * metadata,
    size_t data_size,
    uint32_t hw_buff_sz )
```

Allocate memory for metadata.

Description:

Allocate memory for metadata. The function takes the size of entire input buffer and the data size at which individual block will be compressed. These parameters will be used to calculate and allocate required memory for metadata.

Context:

This function shall not be called in an interrupt context.

Assumptions:

None

Side Effects:

None

Blocking:

Yes

Reentrant:

No

Thread Safe:

Yes

Parameters

in, out	<i>metadata</i>	Pointer to opaque metadata.
in	<i>data_size</i>	Size of uncompressed buffer.
in	<i>hw_buff_sz</i>	Data size at which individual block will be compressed.

Return values

<i>QZ_OK</i>	Function executed successfully
<i>QZ_FAIL</i>	Function did not succeed
<i>QZ_PARAMS</i>	*metadata is NULL, or data_size is 0, or data_size is greater than 1GB, or incorrect hw_buff_sz.

Precondition

None

Postcondition

None

Note

Only a synchronous version of this function is provided.

See also

None

4.1.5.21 qzFree()

```
void qzFree (
    void * m )
```

Free allocated memory

Description:

Free allocated memory.

Context:

This function shall not be called in an interrupt context.

Assumptions:

None

Side Effects:

None

Blocking:

Yes

Reentrant:

No

Thread Safe:

Yes

Parameters

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

Precondition

None

Postcondition

None

Note

Only a synchronous version of this function is provided.

See also

None

4.1.5.22 qzFreeMetadata()

```
int qzFreeMetadata (
    QzMetadataBlob_T metadata )
```

Free memory allocated for metadata.

Description:

Free memory allocated for metadata.

Context:

This function shall not be called in an interrupt context.

Assumptions:

None

Side Effects:

None

Blocking:

Yes

Reentrant:

No

Thread Safe:

Yes

Parameters

in	<i>metadata</i>	Pointer to opaque metadata.
----	-----------------	-----------------------------

Return values

<i>QZ_OK</i>	Function executed successfully.
<i>QZ_FAIL</i>	Function did not succeed.
<i>QZ_PARAMS</i>	metadata is NULL.

Precondition

None

Postcondition

None

Note

Only a synchronous version of this function is provided.

See also

None

4.1.5.23 qzMemFindAddr()

```
int qzMemFindAddr (
    unsigned char * a )
```

Check whether the address is available

Description:

Check whether the address is available.

Context:

This function shall not be called in an interrupt context.

Assumptions:

None

Side Effects:

None

Blocking:

Yes

Reentrant:

No

Thread Safe:

Yes

Parameters

<code>in</code>	<code>a</code>	Address to be checked
-----------------	----------------	-----------------------

Return values

<code>1</code>	The address is available
<code>0</code>	The address is not available

Precondition

None

Postcondition

None

Note

Only a synchronous version of this function is provided.

See also

None

4.1.5.24 qzCompressStream()

```
int qzCompressStream (
    QzSession_T * sess,
    QzStream_T * strm,
    unsigned int last )
```

Compress data in stream and return checksum

Description:

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.

Context:

This function shall not be called in an interrupt context.

Assumptions:

None

Side Effects:

None

Blocking:

Yes

Reentrant:

No

Thread Safe:

Yes

Parameters

in	<i>sess</i>	Session handle (pointer to opaque instance and session data)
in, out	<i>strm</i>	Stream handle
in	<i>last</i>	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

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

Precondition

None

Postcondition

None

Note

Only a synchronous version of this function is provided.

See also

None

4.1.5.25 qzDecompressStream()

```
int qzDecompressStream (
    QzSession_T * sess,
    QzStream_T * strm,
    unsigned int last )
```

Decompress data in stream and return checksum

Description:

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.

Context:

This function shall not be called in an interrupt context.

Assumptions:

None

Side Effects:

None

Blocking:

Yes

Reentrant:

No

Thread Safe:

Yes

Parameters

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

Return values

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

Precondition

None

Postcondition

None

Note

Only a synchronous version of this function is provided.

See also

None

4.1.5.26 qzEndStream()

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

Terminates a QATzip stream

Description:

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

Context:

This function shall not be called in an interrupt context.

Assumptions:

None

Side Effects:

None

Blocking:

Yes

Reentrant:

No

Thread Safe:

Yes

Parameters

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

Return values

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

Precondition

None

Postcondition

None

Note

Only a synchronous version of this function is provided.

See also

None

4.1.5.27 qzGetSoftwareComponentVersionList()

```
int qzGetSoftwareComponentVersionList (
    QzSoftwareVersionInfo_T * api_info,
    unsigned int * num_elem )
```

Requests the release versions of the QATZip Library sub components.

Description:

Populate an array of pre-allocated QzSoftwareVersionInfo_T structs with the names and versions of QATzip sub components.

Context:

This function shall not be called in an interrupt context.

Assumptions:

None

Side Effects:

None

Blocking:

Yes

Reentrant:

Yes

Thread Safe:

Yes

Parameters

in, out	api_info	pointer to a QzSoftwareVersionInfo_T structure to populate.
in, out	num_elem	pointer to an unsigned int expressing how many elements are in the array provided in api_info

Return values

QZ_OK	Function executed successfully
QZ_FAIL	Function did not succeed
QZ_NO_SW_AVAIL	Function did not find a software provider for fallback
QZ_NO_HW	Function did not find an installed kernel driver

Return values

<code>QZ_NOSW_NO_HW</code>	Functions did not find an installed kernel driver or software provider
<code>QZ_PARAMS</code>	*api_info or num_elem is NULL or not large enough to store all QzSoftwareVersionInfo_T structures

Precondition

None

Postcondition

None

Note

Only a synchronous version of this function is provided.

See also

None

4.1.5.28 qzGetSoftwareComponentCount()

```
int qzGetSoftwareComponentCount (
    unsigned int * num_elem )
```

Requests the number of Software components used by the QATZip library

Description:

This function populates num_elem variable with the number of software components available to the library.

Context:

This function shall not be called in an interrupt context.

Assumptions:

None

Side Effects:

None

Blocking:

Yes

Reentrant:

Yes

Thread Safe:

Yes

Parameters

<code>in, out</code>	<code>num_elem</code>	pointer to an unsigned int to populate how many software componets are associated with QATZip
----------------------	-----------------------	---

Return values

<code>QZ_OK</code>	Function executed successfully
<code>QZ_FAIL</code>	Function did not succeed
<code>QZ_NO_SW_AVAIL</code>	Function did not find a software provider for fallback
<code>QZ_NO_HW</code>	Function did not find an installed kernel driver
<code>QZ_NOSW_NO_HW</code>	Functions did not find an installed kernel driver or software provider
<code>QZ_PARAMS</code>	*num_elem is NULL

Precondition

None

Postcondition

None

Note

Only a synchronous version of this function is provided.

See also

None

4.1.5.29 qzGetSessionCrc64Config()

```
int qzGetSessionCrc64Config (  
    QzSession_T * sess,  
    QzCrc64Config_T * crc64_config )
```

Requests the CRC64 configuration of the provided session

Description:

This function populates `crc64_config` with the CRC64 configuration details of `sess`. This function has a dependency on invoking a setup session function first.

Context:

This function shall not be called in an interrupt context.

Assumptions:

None

Side Effects:

None

Blocking:

Yes

Reentrant:

Yes

Thread Safe:

Yes

Parameters

in	<i>sess</i>	Session handle (pointer to opaque instance and session data)
out	<i>crc64_config</i>	Configuration for CRC 64 generation.

Return values

<i>QZ_OK</i>	Function executed successfully
<i>QZ_FAIL</i>	Session was not setup
<i>QZ_PARAMS</i>	*sess or *crc64_config is NULL

Precondition

None

Postcondition

None

Note

Only a synchronous version of this function is provided.

See also

None

4.1.5.30 qzGetSessionCrc32Config()

```
int qzGetSessionCrc32Config (
    QzSession_T * sess,
    QzCrc32Config_T * crc32_config )
```

Requests the CRC32 configuration of the provided session

Description:

This function populates `crc32_config` with the CRC32 configuration details of `sess`. This function has a dependency on invoking a setup session function first.

Context:

This function shall not be called in an interrupt context.

Assumptions:

None

Side Effects:

None

Blocking:

Yes

Reentrant:

Yes

Thread Safe:

Yes

Parameters

in	<code>sess</code>	Session handle (pointer to opaque instance and session data)
out	<code>crc32_config</code>	Configuration for CRC32 generation.

Return values

<code>QZ_OK</code>	Function executed successfully
<code>QZ_FAIL</code>	Session was not setup
<code>QZ_PARAMS</code>	*sess or *crc32_config is NULL

Precondition

None

Postcondition

None

Note

Only a synchronous version of this function is provided.

See also

None

4.1.5.31 qzSetSessionCrc64Config()

```
int qzSetSessionCrc64Config (
    QzSession_T * sess,
    QzCrc64Config_T * crc64_config )
```

Sets the CRC64 configuration of the provided session with a user defined set of parameters.

Description:

This function populates the CRC64 configuration details of `sess` using the parameters provided in `crc64_config`. This function has a dependency on invoking a setup session function first.

Context:

This function shall not be called in an interrupt context.

Assumptions:

None

Side Effects:

None

Blocking:

Yes

Reentrant:

Yes

Thread Safe:

Yes

Parameters

in	<i>sess</i>	Session handle (pointer to opaque instance and session data)
in	<i>crc64_config</i>	Configuration for CRC64 generation.

Return values

<i>QZ_OK</i>	Function executed successfully
<i>QZ_FAIL</i>	Session was not setup
<i>QZ_PARAMS</i>	*sess or *crc64_config is NULL or contains invalid paramters.

Precondition

None

Postcondition

None

Note

Only a synchronous version of this function is provided.

See also

None

4.1.5.32 qzSetSessionCrc32Config()

```
int qzSetSessionCrc32Config (  
    QzSession_T * sess,  
    QzCrc32Config_T * crc32_config )
```

Sets the CRC32 configuration of the provided session with a user defined set of parameters.

Description:

This function populates the CRC32 configuration details of *sess* using the paramaters provided in *crc32_config*. This function has a dependency on invoking a setup session function first.

Context:

This function shall not be called in an interrupt context.

Assumptions:

None

Side Effects:

None

Blocking:

Yes

Reentrant:

Yes

Thread Safe:

Yes

Parameters

in	<i>sess</i>	Session handle (pointer to opaque instance and session data)
in	<i>crc32_config</i>	Configuration for CRC32 generation.

Return values

<i>QZ_OK</i>	Function executed successfully
<i>QZ_FAIL</i>	Session was not setup
<i>QZ_PARAMS</i>	*sess or *crc32_config is NULL or contains invalid parameters.

Precondition

None

Postcondition

None

Note

Only a synchronous version of this function is provided.

See also

None

4.1.5.33 qzMetadataBlockRead()

```
int qzMetadataBlockRead (
    uint32_t block_num,
    QzMetadataBlob_T metadata,
    uint32_t * block_offset,
    uint32_t * block_size,
    uint32_t * block_flags,
    uint32_t * block_hash )
```

Read metadata parameters.

Description:

This function reads metadata information for the block specified by the function param `block_num`.

`block_offset` returns offset value in bytes from the previous compressed block of the compressed data.

`block_size` returns the block size in bytes of the compressed block. Some blocks may be uncompressed if size > threshold as specified during compression and the size returned will reflect the same.

`block_flags` returns the value 1 if the data is compressed and 0 if the data is not compressed.

`block_hash` returns the xxHash value of the plain text of the `hw_buff_sz` payload sent for compression operation.

If NULL is specified for any of the metadata parameters (`block_offset`, `block_size`, `block_flags`, `block_hash`) reading the parameter value will be ignored.

Context:

This function shall not be called in an interrupt context.

Assumptions:

None

Side Effects:

None

Blocking:

Yes

Reentrant:

No

Thread Safe:

Yes

Parameters

in	<i>block_num</i>	Block number of which metadata information should be read.
in	<i>metadata</i>	Pointer to opaque metadata.
in, out	<i>block_offset</i>	Pointer to the block offset value.
in, out	<i>block_size</i>	Pointer to the block size value.
in, out	<i>block_flags</i>	Pointer to the block flags value.
in, out	<i>block_hash</i>	Pointer to the block xxHash value.

Return values

<i>QZ_OK</i>	Function executed successfully.
<i>QZ_FAIL</i>	Function did not succeed.
<i>QZ_PARAMS</i>	Metadata is NULL.
<i>QZ_OUT_OF_RANGE</i>	<i>block_num</i> specified is out of range.

Precondition

None

Postcondition

None

Note

Only a synchronous version of this function is provided.

See also

None

4.1.5.34 qzMetadataBlockWrite()

```
int qzMetadataBlockWrite (
    uint32_t block_num,
    QzMetadataBlob_T metadata,
    uint32_t * block_offset,
    uint32_t * block_size,
    uint32_t * block_flags,
    uint32_t * block_hash )
```

Write metadata parameters.

Description:

This function writes metadata information for the block specified by the function param `block_num`.

`block_offset` writes offset value in bytes from the previous compressed block of the compressed data.

`block_size` writes the block size in bytes of the compressed block.

`block_flags` causes the metadata to indicate the data is compressed if passed a value of 1 and indicates uncompressed if value passed is zero (0).

`block_hash` writes the xxHash value of the plain text of the `hw_buff_sz` payload sent for compression operation.

If NULL is specified for any of the metadata parameters (`block_offset`, `block_size`, `block_flags`, `block_hash`) writing the parameter value into metadata will be ignored.

Context:

This function shall not be called in an interrupt context.

Assumptions:

None

Side Effects:

None

Blocking:

Yes

Reentrant:

No

Thread Safe:

Yes

Parameters

in	<i>block_num</i>	Block number into which metadata information should be written.
in, out	<i>metadata</i>	Pointer to opaque metadata.
in	<i>block_offset</i>	Pointer to the block offset value.
in	<i>block_size</i>	Pointer to the block size value.
in	<i>block_flags</i>	Pointer to the block flags value.
in	<i>block_hash</i>	Pointer to the block xxHash value.

Return values

<i>QZ_OK</i>	Function executed successfully.
<i>QZ_FAIL</i>	Function did not succeed.
<i>QZ_PARAMS</i>	Metadata is NULL.
<i>QZ_OUT_OF_RANGE</i>	block_num specified is out of range.

Precondition

None

Postcondition

None

Note

Only a synchronous version of this function is provided.

See also

None

4.1.5.35 qzMetadataBlockGetCrc64()

```
int qzMetadataBlockGetCrc64 (
    uint32_t block_num,
    QzMetadataBlob_T metadata,
    uint64_t * input_crc,
    uint64_t * output_crc )
```

Read CRC64 of a metadata block.

Description:

This function reads the input and output CRC64 for the block specified by the function param block_num.

Context:

This function shall not be called in an interrupt context.

Assumptions:

None

Side Effects:

None

Blocking:

Yes

Reentrant:

No

Thread Safe:

Yes

Parameters

in	<i>block_num</i>	Block number of which CRC64 should be read.
in	<i>metadata</i>	Pointer to opaque metadata.
out	<i>input_crc</i>	Pointer to the CRC64 calculation on input buffer.
out	<i>output_crc</i>	Pointer to the CRC64 calculation on output buffer.

Return values

<i>QZ_OK</i>	Function executed successfully.
<i>QZ_FAIL</i>	Function did not succeed.
<i>QZ_PARAMS</i>	Metadata is NULL or checksum mismatch.
<i>QZ_OUT_OF_RANGE</i>	block_num specified is out of range.

Precondition

None

Postcondition

None

Note

Only a synchronous version of this function is provided.

See also

None

4.1.5.36 qzMetadataBlockGetCrc32()

```
int qzMetadataBlockGetCrc32 (
    uint32_t block_num,
    QzMetadataBlob_T metadata,
    uint32_t * input_crc,
    uint32_t * output_crc )
```

Read CRC32 of a metadata block.

Description:

This function reads the input and output CRC32 for the block specified by the function param `block_num`.

Context:

This function shall not be called in an interrupt context.

Assumptions:

None

Side Effects:

None

Blocking:

Yes

Reentrant:

No

Thread Safe:

Yes

Parameters

in	<i>block_num</i>	Block number of which CRC32 should be read.
in	<i>metadata</i>	Pointer to opaque metadata.
out	<i>input_crc</i>	Pointer to the CRC32 calculation on input buffer.
out	<i>output_crc</i>	Pointer to the CRC32 calculation on output buffer.

Return values

<i>QZ_OK</i>	Function executed successfully.
<i>QZ_FAIL</i>	Function did not succeed.
<i>QZ_PARAMS</i>	Metadata is NULL or checksum mismatch.
<i>QZ_OUT_OF_RANGE</i>	block_num specified is out of range.

Precondition

None

Postcondition

None

Note

Only a synchronous version of this function is provided.

See also

None

Chapter 5

Data Structure Documentation

5.1 QzCrc32Config_T Struct Reference

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

Data Fields

- uint32_t [polynomial](#)
- uint32_t [initial_value](#)
- uint32_t [reflect_in](#)
- uint32_t [reflect_out](#)
- uint32_t [xor_out](#)

5.1.1 Detailed Description

QATzip CRC32 configuration structure

Description:

This structure contains data relating to configuration of the session's CRC32 functionality.

5.1.2 Field Documentation

5.1.2.1 polynomial

```
uint32_t QzCrc32Config_T::polynomial
```

Polynomial used for CRC32 calculation.

5.1.2.2 initial_value

```
uint32_t QzCrc32Config_T::initial_value
```

Initial CRC32 value.

5.1.2.3 reflect_in

```
uint32_t QzCrc32Config_T::reflect_in
```

Reflect bit order before CRC calculation.

5.1.2.4 reflect_out

```
uint32_t QzCrc32Config_T::reflect_out
```

Reflect bit order after CRC calculation.

5.1.2.5 xor_out

```
uint32_t QzCrc32Config_T::xor_out
```

XOR out value for CRC32 calculation.

5.2 QzCrc64Config_T Struct Reference

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

Data Fields

- uint64_t [polynomial](#)
- uint64_t [initial_value](#)
- uint32_t [reflect_in](#)
- uint32_t [reflect_out](#)
- uint64_t [xor_out](#)

5.2.1 Detailed Description

QATzip CRC64 configuration structure

Description:

This structure contains data relating to configuration of the sessions CRC64 functionality. Session defaults to using ECMA-182 Normal on creation.

5.2.2 Field Documentation

5.2.2.1 polynomial

`uint64_t QzCrc64Config_T::polynomial`

Polynomial used for CRC64 calculation. Default 0x42F0E1EBA9EA3693

5.2.2.2 initial_value

`uint64_t QzCrc64Config_T::initial_value`

Defaults to 0x0000000000000000

5.2.2.3 reflect_in

`uint32_t QzCrc64Config_T::reflect_in`

Reflect bit order before CRC calculation. Default 0

5.2.2.4 reflect_out

`uint32_t QzCrc64Config_T::reflect_out`

Reflect bit order after CRC calculation. Default 0

5.2.2.5 xor_out

`uint64_t QzCrc64Config_T::xor_out`

Defaults to 0x0000000000000000

5.3 QzCrcResult_T Struct Reference

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

Data Fields

- `int status`
- `uint32_t valid_flags`
- `union {
 uint32_t * crc_32
 uint64_t * crc_64
} in_crc`
- `union {
 uint32_t * crc_32
 uint64_t * crc_64
} out_crc`

5.3.1 Detailed Description

crc structure in [QzResult_T](#)

Description:

This structure include the crc input/output value data.
And crc calculate status, whether input/output valid.

5.3.2 Field Documentation

5.3.2.1 status

```
int QzCrcResult_T::status
```

indicate checksum compute's status

5.3.2.2 valid_flags

```
uint32_t QzCrcResult_T::valid_flags
```

indicate whether input and output crc's are valid

5.3.2.3 crc_32

```
uint32_t* QzCrcResult_T::crc_32
```

5.3.2.4 crc_64

```
uint64_t* QzCrcResult_T::crc_64
```

5.3.2.5 [union]

```
union { ... } QzCrcResult_T::in_crc
```

indicate input crc value

5.3.2.6 [union]

```
union { ... } QzCrcResult_T::out_crc
```

indicate output crc value

5.4 QzDictionaryParams_T Struct Reference

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

Data Fields

- const unsigned char * [dictionary](#)
- unsigned int [dictionary_len](#)
- int [compressed](#)
- unsigned long * [crc64Addr](#)
- unsigned int [dictId](#)

5.4.1 Field Documentation

5.4.1.1 dictionary

```
const unsigned char* QzDictionaryParams_T::dictionary
```

5.4.1.2 dictionary_len

```
unsigned int QzDictionaryParams_T::dictionary_len
```

5.4.1.3 compressed

```
int QzDictionaryParams_T::compressed
```

5.4.1.4 crc64Addr

```
unsigned long* QzDictionaryParams_T::crc64Addr
```

5.4.1.5 dictId

```
unsigned int QzDictionaryParams_T::dictId
```

5.5 QzResult_T Struct Reference

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

Data Fields

- int [status](#)
- void * [cb_tag](#)
- unsigned int [src_len](#)
- unsigned int [dest_len](#)
- uint64_t [ext_rc](#)
- [QzCrcResult_T](#) * [crc](#)
- void * [extension_result](#)

5.5.1 Detailed Description

QzResult structure for compress2/decompress2 API

Description:

This structure contains status, custom callback params for async callback function, src_len for input buf length and dest_len for output buf length, and extend return code for post processing and CRC input/output result.

5.5.2 Field Documentation

5.5.2.1 status

```
int QzResult_T::status
```

Status of the offload operation, output only

5.5.2.2 cb_tag

```
void* QzResult_T::cb_tag
```

Opaque to QATzip, Users could uses this pointer for tracking customer side job structures, Most likely, the caller could track other variables including the pointers to the src and dest in
async mode, could set to NULL

5.5.2.3 src_len

```
unsigned int QzResult_T::src_len
```

Length of source buffer. Modified to number of bytes consumed

5.5.2.4 dest_len

```
unsigned int QzResult_T::dest_len
```

Length of destination buffer. Modified to length of produced data when function returns

5.5.2.5 ext_rc

```
uint64_t QzResult_T::ext_rc
```

Extended return codes, output only

5.5.2.6 crc

```
QzCrcResult_T* QzResult_T::crc
```

crc result include input/output, could be NULL

5.5.2.7 extension_result

```
void* QzResult_T::extension_result
```

This pointer is for extensibility of QzResult, please don't use this pointer and always set this pointer as NULL

5.6 QzSession_T Struct Reference

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

Data Fields

- signed long int [hw_session_stat](#)
- int [thd_sess_stat](#)
- void * [internal](#)
- unsigned long [total_in](#)
- unsigned long [total_out](#)

5.6.1 Detailed Description

QATzip Session opaque data storage

Description:

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

5.6.2 Field Documentation

5.6.2.1 hw_session_stat

```
signed long int QzSession_T::hw_session_stat
```

Filled in during initialization, session startup and decompression

5.6.2.2 thd_sess_stat

```
int QzSession_T::thd_sess_stat
```

Note process compression and decompression thread state

5.6.2.3 internal

```
void* QzSession_T::internal
```

Session data is opaque to outside world

5.6.2.4 total_in

```
unsigned long QzSession_T::total_in
```

Total processed input data length in this session

5.6.2.5 total_out

```
unsigned long QzSession_T::total_out
```

Total output data length in this session

5.7 QzSessionParams_T Struct Reference

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

Data Fields

- [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

5.7.1 Detailed Description

QATzip Session Initialization parameters

Description:

This structure contains data for initializing a session.

5.7.2 Field Documentation

5.7.2.1 huffman_hdr

`QzHuffmanHdr_T QzSessionParams_T::huffman_hdr`

Dynamic or Static Huffman headers

5.7.2.2 direction

`QzDirection_T QzSessionParams_T::direction`

Compress or decompress

5.7.2.3 data_fmt

`QzDataFormat_T QzSessionParams_T::data_fmt`

Deflate, deflate with GZip or deflate with GZip ext

5.7.2.4 comp_lvl

`unsigned int QzSessionParams_T::comp_lvl`

Compression level 1 to 9

5.7.2.5 comp_algorithm

`unsigned char QzSessionParams_T::comp_algorithm`

Compress/decompression algorithms

5.7.2.6 max_forks

`unsigned int QzSessionParams_T::max_forks`

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

5.7.2.7 sw_backup

```
unsigned char QzSessionParams_T::sw_backup
```

bit field defining SW configuration (see QZ_SW_* definitions)

5.7.2.8 hw_buff_sz

```
unsigned int QzSessionParams_T::hw_buff_sz
```

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

5.7.2.9 strm_buff_sz

```
unsigned int QzSessionParams_T::strm_buff_sz
```

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

5.7.2.10 input_sz_thrshold

```
unsigned int QzSessionParams_T::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

5.7.2.11 req_cnt_thrshold

```
unsigned int QzSessionParams_T::req_cnt_thrshold
```

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

5.7.2.12 wait_cnt_thrshold

```
unsigned int QzSessionParams_T::wait_cnt_thrshold
```

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

5.8 QzSessionParamsCommon_T Struct Reference

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

Data Fields

- [QzDirection_T](#) direction
- 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](#)
- [QzPollingMode_T](#) polling_mode
- unsigned int [is_sensitive_mode](#)

5.8.1 Field Documentation

5.8.1.1 direction

```
QzDirection_T QzSessionParamsCommon_T::direction
```

Compress or decompress

5.8.1.2 comp_lvl

```
unsigned int QzSessionParamsCommon_T::comp_lvl
```

Compression level 1 to 9

5.8.1.3 comp_algorithm

```
unsigned char QzSessionParamsCommon_T::comp_algorithm
```

Compress/decompression algorithms

5.8.1.4 max_forks

```
unsigned int QzSessionParamsCommon_T::max_forks
```

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

5.8.1.5 sw_backup

```
unsigned char QzSessionParamsCommon_T::sw_backup
```

bit field defining SW configuration (see QZ_SW_* definitions)

5.8.1.6 hw_buff_sz

```
unsigned int QzSessionParamsCommon_T::hw_buff_sz
```

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

5.8.1.7 strm_buff_sz

```
unsigned int QzSessionParamsCommon_T::strm_buff_sz
```

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

5.8.1.8 input_sz_thrshold

```
unsigned int QzSessionParamsCommon_T::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

5.8.1.9 req_cnt_thrshold

```
unsigned int QzSessionParamsCommon_T::req_cnt_thrshold
```

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

5.8.1.10 wait_cnt_thrshold

```
unsigned int QzSessionParamsCommon_T::wait_cnt_thrshold
```

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

5.8.1.11 polling_mode

```
QzPollingMode_T QzSessionParamsCommon_T::polling_mode
```

0 means no busy polling, 1 means busy polling

5.8.1.12 is_sensitive_mode

```
unsigned int QzSessionParamsCommon_T::is_sensitive_mode
```

0 means disable sensitive mode, 1 means enable sensitive mode

5.9 QzSessionParamsDeflate_T Struct Reference

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


Data Fields

- [QzSessionParamsCommon_T](#) `common_params`
- [QzHuffmanHdr_T](#) `huffman_hdr`
- [QzDataFormat_T](#) `data_fmt`

5.9.1 Field Documentation

5.9.1.1 common_params

[QzSessionParamsCommon_T](#) `QzSessionParamsDeflate_T::common_params`

5.9.1.2 huffman_hdr

[QzHuffmanHdr_T](#) `QzSessionParamsDeflate_T::huffman_hdr`

Dynamic or Static Huffman headers

5.9.1.3 data_fmt

[QzDataFormat_T](#) `QzSessionParamsDeflate_T::data_fmt`

Deflate, deflate with GZip or deflate with GZip ext

5.10 QzSessionParamsDeflateExt_T Struct Reference

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

Data Fields

- [QzSessionParamsDeflate_T](#) `deflate_params`
- unsigned char `stop_decompression_stream_end`
- unsigned char `zlib_format`

5.10.1 Field Documentation

5.10.1.1 deflate_params

[QzSessionParamsDeflate_T](#) `QzSessionParamsDeflateExt_T::deflate_params`

5.10.1.2 stop_decompression_stream_end

unsigned char `QzSessionParamsDeflateExt_T::stop_decompression_stream_end`

zlib format, default 0 if zlib format value will be 1

5.10.1.3 `zlib_format`

```
unsigned char QzSessionParamsDeflateExt_T::zlib_format
```

5.11 `QzSessionParamsLZ4_T` Struct Reference

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

Data Fields

- [QzSessionParamsCommon_T](#) `common_params`

5.11.1 Field Documentation

5.11.1.1 `common_params`

```
QzSessionParamsCommon\_T QzSessionParamsLZ4_T::common_params
```

5.12 `QzSessionParamsLZ4S_T` Struct Reference

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

Data Fields

- [QzSessionParamsCommon_T](#) `common_params`
- [qzLZ4SCallbackFn](#) `qzCallback`
- `void *` [qzCallback_external](#)
- `unsigned int` [lz4s_mini_match](#)

5.12.1 Field Documentation

5.12.1.1 `common_params`

```
QzSessionParamsCommon\_T QzSessionParamsLZ4S_T::common_params
```

5.12.1.2 `qzCallback`

```
qzLZ4SCallbackFn QzSessionParamsLZ4S_T::qzCallback
```

post processing callback for zstd compression

5.12.1.3 qzCallback_external

```
void* QzSessionParamsLZ4S_T::qzCallback_external
```

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

5.12.1.4 lz4s_mini_match

```
unsigned int QzSessionParamsLZ4S_T::lz4s_mini_match
```

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

5.13 QzSessionParamsZstd_T Struct Reference

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

Data Fields

- [QzSessionParamsCommon_T](#) common_params
- unsigned int [historyBuffer](#)
- unsigned char [blockSize](#)
- unsigned char [reserved](#) [7]

5.13.1 Field Documentation

5.13.1.1 common_params

```
QzSessionParamsCommon\_T QzSessionParamsZstd_T::common_params
```

5.13.1.2 historyBuffer

```
unsigned int QzSessionParamsZstd_T::historyBuffer
```

5.13.1.3 blockSize

```
unsigned char QzSessionParamsZstd_T::blockSize
```

5.13.1.4 reserved

```
unsigned char QzSessionParamsZstd_T::reserved[7]
```

5.14 QzSoftwareVersionInfo_T Struct Reference

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

Data Fields

- [QzSoftwareComponentType_T](#) `component_type`
- unsigned char `component_name` [[QZ_MAX_STRING_LENGTH](#)]
- unsigned int `major_version`
- unsigned int `minor_version`
- unsigned int `patch_version`
- unsigned int `build_number`
- unsigned char `reserved` [52]

5.14.1 Field Documentation

5.14.1.1 `component_type`

```
QzSoftwareComponentType\_T QzSoftwareVersionInfo_T::component_type
```

5.14.1.2 `component_name`

```
unsigned char QzSoftwareVersionInfo_T::component_name [QZ\_MAX\_STRING\_LENGTH]
```

5.14.1.3 `major_version`

```
unsigned int QzSoftwareVersionInfo_T::major_version
```

5.14.1.4 `minor_version`

```
unsigned int QzSoftwareVersionInfo_T::minor_version
```

5.14.1.5 `patch_version`

```
unsigned int QzSoftwareVersionInfo_T::patch_version
```

5.14.1.6 `build_number`

```
unsigned int QzSoftwareVersionInfo_T::build_number
```

5.14.1.7 `reserved`

```
unsigned char QzSoftwareVersionInfo_T::reserved[52]
```

5.15 QzStatus_T Struct Reference

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

Data Fields

- 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_allocated](#)
- unsigned char [using_huge_pages](#)
- signed long int [hw_session_status](#)
- unsigned char [algo_sw](#) [QZ_MAX_ALGORITHMS]
- unsigned char [algo_hw](#) [QZ_MAX_ALGORITHMS]

5.15.1 Detailed Description

QATzip status structure

Description:

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

5.15.2 Field Documentation

5.15.2.1 qat_hw_count

```
unsigned short int QzStatus_T::qat_hw_count
```

From PCI scan

5.15.2.2 qat_service_init

```
unsigned char QzStatus_T::qat_service_init
```

Check if the available services have been initialized

5.15.2.3 qat_mem_drvr

```
unsigned char QzStatus_T::qat_mem_drvr
```

1 if /dev/qat_mem exists 2 if /dev/qat_mem has been opened 0 otherwise

5.15.2.4 qat_instance_attach

```
unsigned char QzStatus_T::qat_instance_attach
```

Is this thread/g_process properly attached to an Instance?

5.15.2.5 memory_allocated

```
unsigned long int QzStatus_T::memory_allocated
```

Amount of memory allocated by this thread/process

5.15.2.6 using_huge_pages

```
unsigned char QzStatus_T::using_huge_pages
```

Are memory slabs coming from huge pages?

5.15.2.7 hw_session_status

```
signed long int QzStatus_T::hw_session_status
```

One of QATzip Session Status

5.15.2.8 algo_sw

```
unsigned char QzStatus_T::algo_sw[QZ_MAX_ALGORITHMS]
```

Support software algorithms

5.15.2.9 algo_hw

```
unsigned char QzStatus_T::algo_hw[QZ_MAX_ALGORITHMS]
```

Count of hardware devices supporting algorithms

5.16 QzStream_T Struct Reference

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

Data Fields

- 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](#)

5.16.1 Detailed Description

QATzip Stream data storage

Description:

This structure contains metadata needed for stream operation.

5.16.2 Field Documentation

5.16.2.1 in_sz

```
unsigned int QzStream_T::in_sz
```

Set by application, reset by QATzip to indicate consumed data

5.16.2.2 out_sz

```
unsigned int QzStream_T::out_sz
```

Set by application, reset by QATzip to indicate processed data

5.16.2.3 in

```
unsigned char* QzStream_T::in
```

Input data pointer set by application

5.16.2.4 out

```
unsigned char* QzStream_T::out
```

Output data pointer set by application

5.16.2.5 pending_in

```
unsigned int QzStream_T::pending_in
```

Unprocessed bytes held in QATzip

5.16.2.6 pending_out

```
unsigned int QzStream_T::pending_out
```

Processed bytes held in QATzip

5.16.2.7 crc_type

```
QzCrcType_T QzStream_T::crc_type
```

Checksum type in Adler, CRC32 or none

5.16.2.8 crc_32

```
unsigned int QzStream_T::crc_32
```

Checksum value

5.16.2.9 reserved

```
unsigned long long QzStream_T::reserved
```

Reserved for future use

5.16.2.10 opaque

```
void* QzStream_T::opaque
```

Internal storage managed by QATzip

Chapter 6

File Documentation

6.1 qatzip.h File Reference

Data Structures

- struct [QzSessionParams_T](#)
- struct [QzSessionParamsCommon_T](#)
- struct [QzSessionParamsDeflate_T](#)
- struct [QzSessionParamsLZ4_T](#)
- struct [QzSessionParamsLZ4S_T](#)
- struct [QzSessionParamsZstd_T](#)
- struct [QzDictionaryParams_T](#)
- struct [QzSessionParamsDeflateExt_T](#)
- struct [QzSession_T](#)
- struct [QzStatus_T](#)
- struct [QzSoftwareVersionInfo_T](#)
- struct [QzCrc64Config_T](#)
- struct [QzCrc32Config_T](#)
- struct [QzCrcResult_T](#)
- struct [QzResult_T](#)
- struct [QzStream_T](#)

Macros

- #define [QATZIP_API_VERSION_NUM_MAJOR](#) (2)
- #define [QATZIP_API_VERSION_NUM_MINOR](#) (6)
- #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_METADATA_OVERFLOW (-118)`
- `#define QZ_OUT_OF_RANGE (-119)`
- `#define QZ_NOT_SUPPORTED (-200)`
- `#define QZ_MAX_ALGORITHMS ((int)255)`
- `#define QZ_DEFLATE ((unsigned char)8)`
- `#define QZ_LZ4 ((unsigned char)'4')`
- `#define QZ_LZ4_BLOCK ((unsigned char)'B')`
- `#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_Gen3 (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_Gen3 (2*1024*1024*1024U)`
- `#define QZ_STRM_BUFF_SZ_DEFAULT QZ_HW_BUFF_SZ`
- `#define QZ_STRM_BUFF_MIN_SZ (1*1024)`
- `#define QZ_STRM_BUFF_MAX_SZ (2*1024*1024 - 5*1024)`
- `#define QZ_COMP_THRESHOLD_DEFAULT 1024`
- `#define QZ_COMP_THRESHOLD_MINIMUM 128`
- `#define QZ_REQ_THRESHOLD_MINIMUM 1`
- `#define QZ_REQ_THRESHOLD_MAXIMUM NUM_BUFF`
- `#define QZ_REQ_THRESHOLD_DEFAULT QZ_REQ_THRESHOLD_MAXIMUM`
- `#define QZ_WAIT_CNT_THRESHOLD_DEFAULT 8`
- `#define QZ_DEFLATE_COMP_LVL_MINIMUM (1)`
- `#define QZ_DEFLATE_COMP_LVL_MAXIMUM (9)`
- `#define QZ_DEFLATE_COMP_LVL_MAXIMUM_Gen3 (12)`
- `#define QZ_LZS_COMP_LVL_MINIMUM (1)`
- `#define QZ_LZS_COMP_LVL_MAXIMUM (12)`
- `#define QZ_AUTO_SELECT_NUMA_NODE (-1)`
- `#define QZ_ZSTD_COMP_LVL_MINIMUM (1)`
- `#define QZ_ZSTD_COMP_LVL_MAXIMUM (12)`
- `#define QZ_ZSTD_HIST_BUFF_DEFAULT (64*1024)`

- `#define QZ_ZSTD_BLOCKSIZE_DEFAULT (128)`
- `#define QZ_ZSTD_HW_BUFF_SZ (128*1024)`
- `#define QZ_SW_BACKUP_BIT_POSITION (0)`
- `#define QZ_SW_FORCESW_BIT_POSITION (1)`
- `#define QZ_ENABLE_SOFTWARE_BACKUP(_BackupVariable) (_BackupVariable |= (1 << QZ_SW_BACKUP_BIT_POSITION))`
- `#define QZ_ENABLE_SOFTWARE_ONLY_EXECUTION(_BackupVariable) (_BackupVariable |= (1 << QZ_SW_FORCESW_BIT_POSITION))`
- `#define QZ_DISABLE_SOFTWARE_BACKUP(_BackupVariable) (_BackupVariable &= ~(1 << QZ_SW_BACKUP_BIT_POSITION))`
- `#define QZ_DISABLE_SOFTWARE_ONLY_EXECUTION(_BackupVariable) (_BackupVariable &= ~(1 << QZ_SW_FORCESW_BIT_POSITION))`
- `#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_POST_PROCESS_FAIL_BIT (10)`
- `#define QZ_POST_PROCESS_FAIL_MASK (1 << QZ_POST_PROCESS_FAIL_BIT)`
- `#define QZ_POST_PROCESS_FAIL(ret, ext_rc) (ret && (ext_rc & QZ_POST_PROCESS_FAIL_MASK))`
- `#define QZ_MAX_STRING_LENGTH 64`
- `#define QZ_INPUT_CRC_VALID_BIT (4)`
- `#define QZ_INPUT_CRC_VALID_MASK (1 << QZ_INPUT_CRC_VALID_BIT)`
- `#define QZ_OUTPUT_CRC_VALID_BIT (8)`
- `#define QZ_OUTPUT_CRC_VALID_MASK (1 << QZ_OUTPUT_CRC_VALID_BIT)`
- `#define QZ_CRC32_VALID_BIT (12)`
- `#define QZ_CRC32_VALID_MASK (1 << QZ_CRC32_VALID_BIT)`
- `#define QZ_CRC64_VALID_BIT (16)`
- `#define QZ_CRC64_VALID_MASK (1 << QZ_CRC64_VALID_BIT)`
- `#define QZ_CRC32_VALID(valid_flags) ((valid_flags & QZ_CRC32_VALID_MASK) && !(valid_flags & QZ_CRC64_VALID_MASK))`
- `#define QZ_CRC64_VALID(valid_flags) ((valid_flags & QZ_CRC64_VALID_MASK) && !(valid_flags & QZ_CRC32_VALID_MASK))`
- `#define QZ_INPUT_CRC_VALID(valid_flags) (valid_flags & QZ_INPUT_CRC_VALID_MASK)`
- `#define QZ_OUTPUT_CRC_VALID(valid_flags) (valid_flags & QZ_OUTPUT_CRC_VALID_BIT)`
- `#define QZ_SKID_PAD_SZ 48`
- `#define QZ_COMPRESSED_SZ_OF_EMPTY_FILE 34`

Typedefs

- `typedef int(* qzLZ4SCallbackFn) (void *external, const unsigned char *src, unsigned int *src_len, unsigned char *dest, unsigned int *dest_len, int *ExtStatus)`
- `typedef int(* qzAsyncCallbackFn) (QzResult_T *res)`
- `typedef void * QzMetadataBlob_T`

Enumerations

- `enum QzHuffmanHdr_T { QZ_DYNAMIC_HDR = 0 , QZ_STATIC_HDR }`
- `enum PinMem_T { COMMON_MEM = 0 , PINNED_MEM }`
- `enum QzDirection_T { QZ_DIR_COMPRESS = 0 , QZ_DIR_DECOMPRESS , QZ_DIR_BOTH }`
- `enum QzDataFormat_T { QZ_DEFLATE_4B = 0 , QZ_DEFLATE_GZIP , QZ_DEFLATE_GZIP_EXT , QZ_DEFLATE_RAW , QZ_FMT_NUM }`
- `enum QzPollingMode_T { QZ_PERIODICAL_POLLING = 0 , QZ_BUSY_POLLING }`

- enum [QzCrcType_T](#) { [QZ_CRC32](#) = 0 , [QZ_ADLER](#) , [QZ_XXHASH](#) , [NONE](#) }
- enum [QzSoftwareComponentType_T](#) { [QZ_COMPONENT_FIRMWARE](#) = 0 , [QZ_COMPONENT_KERNEL_DRIVER](#) , [QZ_COMPONENT_USER_DRIVER](#) , [QZ_COMPONENT_QATZIP_API](#) , [QZ_COMPONENT_SOFTWARE_PROVIDER](#) }
- enum [QzLogLevel_T](#) { [LOG_NONE](#) = 0 , [LOG_FATAL](#) , [LOG_ERROR](#) , [LOG_WARNING](#) , [LOG_INFO](#) , [LOG_DEBUG1](#) , [LOG_DEBUG2](#) , [LOG_DEBUG3](#) }

Functions

- [QzLogLevel_T](#) [qzSetLogLevel](#) ([QzLogLevel_T](#) level)
- int [qzInit](#) ([QzSession_T](#) *sess, unsigned char sw_backup)
- int [qzSetupSession](#) ([QzSession_T](#) *sess, [QzSessionParams_T](#) *params)
- int [qzSetupSessionDeflate](#) ([QzSession_T](#) *sess, [QzSessionParamsDeflate_T](#) *params)
- int [qzSetupSessionLZ4](#) ([QzSession_T](#) *sess, [QzSessionParamsLZ4_T](#) *params)
- int [qzSetupSessionLZ4Block](#) ([QzSession_T](#) *sess, [QzSessionParamsLZ4_T](#) *params)
- int [qzSetupSessionLZ4S](#) ([QzSession_T](#) *sess, [QzSessionParamsLZ4S_T](#) *params)
- int [qzSetupSessionZstd](#) ([QzSession_T](#) *sess, [QzSessionParamsZstd_T](#) *params)
- int [qzSetupSessionDeflateExt](#) ([QzSession_T](#) *sess, [QzSessionParamsDeflateExt_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 [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)
- 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 [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)
- int [qzCompressCrc64](#) ([QzSession_T](#) *sess, const unsigned char *src, unsigned int *src_len, unsigned char *dest, unsigned int *dest_len, unsigned int last, uint64_t *crc)
- int [qzCompressCrc64Ext](#) ([QzSession_T](#) *sess, const unsigned char *src, unsigned int *src_len, unsigned char *dest, unsigned int *dest_len, unsigned int last, uint64_t *crc, uint64_t *ext_rc)
- int [qzCompressWithMetadataExt](#) ([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, [QzMetadataBlob_T](#) metadata, uint32_t hw_buff_sz_override, uint32_t comp_thrshld)
- int [qzCompressWithDictionary](#) ([QzSession_T](#) *sess, const unsigned char *src, unsigned char *dest, [QzDictionaryParams_T](#) *dictionary, [qzAsyncCallbackFn](#) callback, [QzResult_T](#) *qzResults)
- int [qzCompress2](#) ([QzSession_T](#) *sess, const unsigned char *src, unsigned char *dest, [qzAsyncCallbackFn](#) callback, [QzResult_T](#) *qzResults)
- int [qzDecompress](#) ([QzSession_T](#) *sess, const unsigned char *src, unsigned int *src_len, unsigned char *dest, unsigned int *dest_len)
- int [qzDecompressExt](#) ([QzSession_T](#) *sess, const unsigned char *src, unsigned int *src_len, unsigned char *dest, unsigned int *dest_len, uint64_t *ext_rc)
- int [qzDecompressCrc](#) ([QzSession_T](#) *sess, const unsigned char *src, unsigned int *src_len, unsigned char *dest, unsigned int *dest_len, unsigned long *crc)
- int [qzDecompressCrcExt](#) ([QzSession_T](#) *sess, const unsigned char *src, unsigned int *src_len, unsigned char *dest, unsigned int *dest_len, unsigned long *crc, uint64_t *ext_rc)
- int [qzDecompressCrc64](#) ([QzSession_T](#) *sess, const unsigned char *src, unsigned int *src_len, unsigned char *dest, unsigned int *dest_len, uint64_t *crc)
- int [qzDecompressCrc64Ext](#) ([QzSession_T](#) *sess, const unsigned char *src, unsigned int *src_len, unsigned char *dest, unsigned int *dest_len, uint64_t *crc, uint64_t *ext_rc)
- int [qzDecompressWithDictionary](#) ([QzSession_T](#) *sess, const unsigned char *src, unsigned char *dest, [QzDictionaryParams_T](#) *dictionary, [qzAsyncCallbackFn](#) callback, [QzResult_T](#) *qzResults)
- int [qzDecompressWithMetadataExt](#) ([QzSession_T](#) *sess, const unsigned char *src, unsigned int *src_len, unsigned char *dest, unsigned int *dest_len, uint64_t *ext_rc, [QzMetadataBlob_T](#) metadata, uint32_t hw_buff_sz_override)

- int [qzDecompress2](#) ([QzSession_T](#) *sess, const unsigned char *src, unsigned char *dest, [qzAsyncCallbackFn](#) callback, [QzResult_T](#) *qzResults)
- int [qzTeardownSession](#) ([QzSession_T](#) *sess)
- int [qzClose](#) ([QzSession_T](#) *sess)
- int [qzGetStatus](#) ([QzSession_T](#) *sess, [QzStatus_T](#) *status)
- int [qzGetDeflateEndOfStream](#) ([QzSession_T](#) *sess, unsigned char *endofstream)
- unsigned int [qzMaxCompressedLength](#) (unsigned int src_sz, [QzSession_T](#) *sess)
- int [qzSetDefaults](#) ([QzSessionParams_T](#) *defaults)
- int [qzSetDefaultsDeflate](#) ([QzSessionParamsDeflate_T](#) *defaults)
- int [qzSetDefaultsLZ4](#) ([QzSessionParamsLZ4_T](#) *defaults)
- int [qzSetDefaultsLZ4Block](#) ([QzSessionParamsLZ4_T](#) *defaults)
- int [qzSetDefaultsLZ4S](#) ([QzSessionParamsLZ4S_T](#) *defaults)
- int [qzSetDefaultsZstd](#) ([QzSessionParamsZstd_T](#) *defaults)
- int [qzSetDefaultsDeflateExt](#) ([QzSessionParamsDeflateExt_T](#) *defaults)
- int [qzGetDefaults](#) ([QzSessionParams_T](#) *defaults)
- int [qzGetDefaultsDeflate](#) ([QzSessionParamsDeflate_T](#) *defaults)
- int [qzGetDefaultsLZ4](#) ([QzSessionParamsLZ4_T](#) *defaults)
- int [qzGetDefaultsLZ4Block](#) ([QzSessionParamsLZ4_T](#) *defaults)
- int [qzGetDefaultsLZ4S](#) ([QzSessionParamsLZ4S_T](#) *defaults)
- int [qzGetDefaultsZstd](#) ([QzSessionParamsZstd_T](#) *defaults)
- int [qzGetDefaultsDeflateExt](#) ([QzSessionParamsDeflateExt_T](#) *defaults)
- void * [qzMalloc](#) (size_t sz, int numa, int force_pinned)
- int [qzAllocateMetadata](#) ([QzMetadataBlob_T](#) *metadata, size_t data_size, uint32_t hw_buff_sz)
- void [qzFree](#) (void *m)
- int [qzFreeMetadata](#) ([QzMetadataBlob_T](#) metadata)
- 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)
- int [qzGetSoftwareComponentVersionList](#) ([QzSoftwareVersionInfo_T](#) *api_info, unsigned int *num_elem)
- int [qzGetSoftwareComponentCount](#) (unsigned int *num_elem)
- int [qzGetSessionCrc64Config](#) ([QzSession_T](#) *sess, [QzCrc64Config_T](#) *crc64_config)
- int [qzGetSessionCrc32Config](#) ([QzSession_T](#) *sess, [QzCrc32Config_T](#) *crc32_config)
- int [qzSetSessionCrc64Config](#) ([QzSession_T](#) *sess, [QzCrc64Config_T](#) *crc64_config)
- int [qzSetSessionCrc32Config](#) ([QzSession_T](#) *sess, [QzCrc32Config_T](#) *crc32_config)
- int [qzMetadataBlockRead](#) (uint32_t block_num, [QzMetadataBlob_T](#) metadata, uint32_t *block_offset, uint32_t *block_size, uint32_t *block_flags, uint32_t *block_hash)
- int [qzMetadataBlockWrite](#) (uint32_t block_num, [QzMetadataBlob_T](#) metadata, uint32_t *block_offset, uint32_t *block_size, uint32_t *block_flags, uint32_t *block_hash)
- int [qzMetadataBlockGetCrc64](#) (uint32_t block_num, [QzMetadataBlob_T](#) metadata, uint64_t *input_crc, uint64_t *output_crc)
- int [qzMetadataBlockGetCrc32](#) (uint32_t block_num, [QzMetadataBlob_T](#) metadata, uint32_t *input_crc, uint32_t *output_crc)

6.1.1 Macro Definition Documentation

6.1.1.1 QATZIP_API_VERSION

```
#define QATZIP_API_VERSION
```

Value:

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

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 QATZIP_BUILD_DLL must be defined when building a DLL No definition required if building the project as static library

6.1.1.3 QZ_DUPLICATE

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

Can not process function again. No failure

6.1.1.4 QZ_FORCE_SW

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

Using SW: Switch to software because of previous block

6.1.1.5 QZ_PARAMS

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

Invalid parameter in function call

6.1.1.6 QZ_FAIL

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

Unspecified error

6.1.1.7 QZ_BUF_ERROR

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

Insufficient buffer error

6.1.1.8 QZ_DATA_ERROR

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

Input data was corrupted

6.1.1.9 QZ_TIMEOUT

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

Operation timed out

6.1.1.10 QZ_INTEG

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

Integrity checked failed

6.1.1.11 QZ_NO_HW

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

Using SW: No QAT HW detected

6.1.1.12 QZ_NO_MDRV

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

Using SW: No memory driver detected

6.1.1.13 QZ_NO_INST_ATTACH

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

Using SW: Could not attach to an instance

6.1.1.14 QZ_LOW_MEM

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

Using SW: Not enough pinned memory

6.1.1.15 QZ_LOW_DEST_MEM

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

Using SW: Not enough pinned memory for dest buffer

6.1.1.16 QZ_UNSUPPORTED_FMT

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

Using SW: QAT device does not support data format

6.1.1.17 QZ_NONE

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

Device uninitialized

6.1.1.18 QZ_NOSW_NO_HW

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

Not using SW: No QAT HW detected

6.1.1.19 QZ_NOSW_NO_MDRV

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

Not using SW: No memory driver detected

6.1.1.20 QZ_NOSW_NO_INST_ATTACH

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

Not using SW: Could not attach to instance

6.1.1.21 QZ_NOSW_LOW_MEM

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

Not using SW: not enough pinned memory

6.1.1.22 QZ_NO_SW_AVAIL

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

Session may require software, but no software is available

6.1.1.23 QZ_NOSW_UNSUPPORTED_FMT

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

Not using SW: QAT device does not support data format

6.1.1.24 QZ_POST_PROCESS_ERROR

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

Using post process: post process callback encountered an error

6.1.1.25 QZ_METADATA_OVERFLOW

```
#define QZ_METADATA_OVERFLOW (-118)
```

Insufficient memory allocated for metadata

6.1.1.26 QZ_OUT_OF_RANGE

```
#define QZ_OUT_OF_RANGE (-119)
```

Metadata block_num specified is out of range

6.1.1.27 QZ_NOT_SUPPORTED

```
#define QZ_NOT_SUPPORTED (-200)
```

Request not supported

6.1.1.28 QZ_MAX_ALGORITHMS

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

6.1.1.29 QZ_DEFLATE

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

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

6.1.1.30 QZ_LZ4

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

6.1.1.31 QZ_LZ4_BLOCK

```
#define QZ_LZ4_BLOCK ((unsigned char)'B')
```

6.1.1.32 QZ_LZ4s

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

6.1.1.33 QZ_ZSTD

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

6.1.1.34 MIN

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

6.1.1.35 QZ_HUFF_HDR_DEFAULT

```
#define QZ_HUFF_HDR_DEFAULT QZ_DYNAMIC_HDR
```

6.1.1.36 QZ_DIRECTION_DEFAULT

```
#define QZ_DIRECTION_DEFAULT QZ_DIR_BOTH
```

6.1.1.37 QZ_DATA_FORMAT_DEFAULT

```
#define QZ_DATA_FORMAT_DEFAULT QZ_DEFLATE_GZIP_EXT
```

6.1.1.38 QZ_COMP_LEVEL_DEFAULT

```
#define QZ_COMP_LEVEL_DEFAULT 1
```

6.1.1.39 QZ_COMP_ALGOL_DEFAULT

```
#define QZ_COMP_ALGOL_DEFAULT QZ_DEFLATE
```

6.1.1.40 QZ_POLL_SLEEP_DEFAULT

```
#define QZ_POLL_SLEEP_DEFAULT 10
```

6.1.1.41 QZ_MAX_FORK_DEFAULT

```
#define QZ_MAX_FORK_DEFAULT 3
```

6.1.1.42 QZ_SW_BACKUP_DEFAULT

```
#define QZ_SW_BACKUP_DEFAULT 1
```

6.1.1.43 QZ_HW_BUFF_SZ

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

6.1.1.44 QZ_HW_BUFF_SZ_Gen3

```
#define QZ_HW_BUFF_SZ_Gen3 (1*1024*1024)
```

6.1.1.45 QZ_HW_BUFF_MIN_SZ

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

6.1.1.46 QZ_HW_BUFF_MAX_SZ

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

6.1.1.47 QZ_HW_BUFF_MAX_SZ_Gen3

```
#define QZ_HW_BUFF_MAX_SZ_Gen3 (2*1024*1024*1024U)
```

6.1.1.48 QZ_STRM_BUFF_SZ_DEFAULT

```
#define QZ_STRM_BUFF_SZ_DEFAULT QZ\_HW\_BUFF\_SZ
```

6.1.1.49 QZ_STRM_BUFF_MIN_SZ

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

6.1.1.50 QZ_STRM_BUFF_MAX_SZ

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

6.1.1.51 QZ_COMP_THRESHOLD_DEFAULT

```
#define QZ_COMP_THRESHOLD_DEFAULT 1024
```

6.1.1.52 QZ_COMP_THRESHOLD_MINIMUM

```
#define QZ_COMP_THRESHOLD_MINIMUM 128
```

6.1.1.53 QZ_REQ_THRESHOLD_MINIMUM

```
#define QZ_REQ_THRESHOLD_MINIMUM 1
```

6.1.1.54 QZ_REQ_THRESHOLD_MAXIMUM

```
#define QZ_REQ_THRESHOLD_MAXIMUM NUM_BUFF
```

6.1.1.55 QZ_REQ_THRESHOLD_DEFAULT

```
#define QZ_REQ_THRESHOLD_DEFAULT QZ\_REQ\_THRESHOLD\_MAXIMUM
```

6.1.1.56 QZ_WAIT_CNT_THRESHOLD_DEFAULT

```
#define QZ_WAIT_CNT_THRESHOLD_DEFAULT 8
```

6.1.1.57 QZ_DEFLATE_COMP_LVL_MINIMUM

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

6.1.1.58 QZ_DEFLATE_COMP_LVL_MAXIMUM

```
#define QZ_DEFLATE_COMP_LVL_MAXIMUM (9)
```

6.1.1.59 QZ_DEFLATE_COMP_LVL_MAXIMUM_Gen3

```
#define QZ_DEFLATE_COMP_LVL_MAXIMUM_Gen3 (12)
```

6.1.1.60 QZ_LZS_COMP_LVL_MINIMUM

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

6.1.1.61 QZ_LZS_COMP_LVL_MAXIMUM

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

6.1.1.62 QZ_AUTO_SELECT_NUMA_NODE

```
#define QZ_AUTO_SELECT_NUMA_NODE (-1)
```

6.1.1.63 QZ_ZSTD_COMP_LVL_MINIMUM

```
#define QZ_ZSTD_COMP_LVL_MINIMUM (1)
```

6.1.1.64 QZ_ZSTD_COMP_LVL_MAXIMUM

```
#define QZ_ZSTD_COMP_LVL_MAXIMUM (12)
```

6.1.1.65 QZ_ZSTD_HIST_BUFF_DEFAULT

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

6.1.1.66 QZ_ZSTD_BLOCKSIZE_DEFAULT

```
#define QZ_ZSTD_BLOCKSIZE_DEFAULT (128)
```

6.1.1.67 QZ_ZSTD_HW_BUFF_SZ

```
#define QZ_ZSTD_HW_BUFF_SZ (128*1024)
```

6.1.1.68 QZ_SW_FORCESW_BIT_POSITION

```
#define QZ_SW_FORCESW_BIT_POSITION (1)
```

6.1.1.69 QZ_ENABLE_SOFTWARE_BACKUP

```
#define QZ_ENABLE_SOFTWARE_BACKUP(  
    _BackupVariable )  (_BackupVariable |= (1 << QZ_SW_BACKUP_BIT_POSITION))
```

6.1.1.70 QZ_ENABLE_SOFTWARE_ONLY_EXECUTION

```
#define QZ_ENABLE_SOFTWARE_ONLY_EXECUTION(  
    _BackupVariable )  (_BackupVariable |= (1 << QZ_SW_FORCESW_BIT_POSITION))
```

< SW backup/fallback enabled

6.1.1.71 QZ_DISABLE_SOFTWARE_BACKUP

```
#define QZ_DISABLE_SOFTWARE_BACKUP(  
    _BackupVariable )  (_BackupVariable &= ~(1 << QZ_SW_BACKUP_BIT_POSITION))
```

< Force SW to perform all compression/decompression operations

6.1.1.72 QZ_DISABLE_SOFTWARE_ONLY_EXECUTION

```
#define QZ_DISABLE_SOFTWARE_ONLY_EXECUTION(  
    _BackupVariable )  (_BackupVariable &= ~(1 << QZ_SW_FORCESW_BIT_POSITION))
```

< SW backup/fallback disabled

6.1.1.73 QZ_SW_EXECUTION_MASK

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

6.1.1.74 QZ_SW_EXECUTION

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

6.1.1.75 QZ_TIMEOUT_BIT

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

6.1.1.76 QZ_TIMEOUT_MASK

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

6.1.1.77 QZ_HW_TIMEOUT

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

6.1.1.78 QZ_POST_PROCESS_FAIL_BIT

```
#define QZ_POST_PROCESS_FAIL_BIT (10)
```

6.1.1.79 QZ_POST_PROCESS_FAIL_MASK

```
#define QZ_POST_PROCESS_FAIL_MASK (1 << QZ_POST_PROCESS_FAIL_BIT)
```

6.1.1.80 QZ_POST_PROCESS_FAIL

```
#define QZ_POST_PROCESS_FAIL(  
    ret,  
    ext_rc )  (ret && (ext_rc & QZ_POST_PROCESS_FAIL_MASK))
```

6.1.1.81 QZ_INPUT_CRC_VALID_MASK

```
#define QZ_INPUT_CRC_VALID_MASK (1 << QZ_INPUT_CRC_VALID_BIT)
```

6.1.1.82 QZ_OUTPUT_CRC_VALID_BIT

```
#define QZ_OUTPUT_CRC_VALID_BIT (8)
```

6.1.1.83 QZ_OUTPUT_CRC_VALID_MASK

```
#define QZ_OUTPUT_CRC_VALID_MASK (1 << QZ_OUTPUT_CRC_VALID_BIT)
```

6.1.1.84 QZ_CRC32_VALID_BIT

```
#define QZ_CRC32_VALID_BIT (12)
```

6.1.1.85 QZ_CRC32_VALID_MASK

```
#define QZ_CRC32_VALID_MASK (1 << QZ_CRC32_VALID_BIT)
```

6.1.1.86 QZ_CRC64_VALID_BIT

```
#define QZ_CRC64_VALID_BIT (16)
```

6.1.1.87 QZ_CRC64_VALID_MASK

```
#define QZ_CRC64_VALID_MASK (1 << QZ_CRC64_VALID_BIT)
```

6.1.1.88 QZ_CRC32_VALID

```
#define QZ_CRC32_VALID(  
    valid_flags )    ((valid_flags & QZ_CRC32_VALID_MASK) && !(valid_flags & QZ_CRC64_VALID_MASK))
```

6.1.1.89 QZ_CRC64_VALID

```
#define QZ_CRC64_VALID(  
    valid_flags )    ((valid_flags & QZ_CRC64_VALID_MASK) && !(valid_flags & QZ_CRC32_VALID_MASK))
```

6.1.1.90 QZ_INPUT_CRC_VALID

```
#define QZ_INPUT_CRC_VALID(  
    valid_flags )    (valid_flags & QZ_INPUT_CRC_VALID_MASK)
```

6.1.1.91 QZ_OUTPUT_CRC_VALID

```
#define QZ_OUTPUT_CRC_VALID(  
    valid_flags )    (valid_flags & QZ_OUTPUT_CRC_VALID_BIT)
```

6.1.1.92 QZ_COMPRESSED_SZ_OF_EMPTY_FILE

```
#define QZ_COMPRESSED_SZ_OF_EMPTY_FILE 34
```

6.1.2 Function Documentation

6.1.2.1 qzSetupSessionDeflate()

```
int qzSetupSessionDeflate (  
    QzSession_T * sess,  
    QzSessionParamsDeflate_T * params )
```

6.1.2.2 qzSetupSessionLZ4()

```
int qzSetupSessionLZ4 (  
    QzSession_T * sess,  
    QzSessionParamsLZ4_T * params )
```

6.1.2.3 qzSetupSessionLZ4Block()

```
int qzSetupSessionLZ4Block (  
    QzSession_T * sess,  
    QzSessionParamsLZ4_T * params )
```

6.1.2.4 qzSetupSessionLZ4S()

```
int qzSetupSessionLZ4S (  
    QzSession_T * sess,  
    QzSessionParamsLZ4S_T * params )
```

6.1.2.5 qzSetupSessionZstd()

```
int qzSetupSessionZstd (  
    QzSession_T * sess,  
    QzSessionParamsZstd_T * params )
```

6.1.2.6 qzSetupSessionDeflateExt()

```
int qzSetupSessionDeflateExt (  
    QzSession_T * sess,  
    QzSessionParamsDeflateExt_T * params )
```


6.1.2.7 qzCompressExt()

```
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 )
```

6.1.2.8 qzCompressCrcExt()

```
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 )
```

6.1.2.9 qzCompressCrc64()

```
int qzCompressCrc64 (
    QzSession_T * sess,
    const unsigned char * src,
    unsigned int * src_len,
    unsigned char * dest,
    unsigned int * dest_len,
    unsigned int last,
    uint64_t * crc )
```

6.1.2.10 qzCompressCrc64Ext()

```
int qzCompressCrc64Ext (
    QzSession_T * sess,
    const unsigned char * src,
    unsigned int * src_len,
    unsigned char * dest,
    unsigned int * dest_len,
    unsigned int last,
    uint64_t * crc,
    uint64_t * ext_rc )
```

6.1.2.11 qzDecompressExt()

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

6.1.2.12 qzDecompressCrcExt()

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

6.1.2.13 qzDecompressCrc64()

```
int qzDecompressCrc64 (
    QzSession_T * sess,
    const unsigned char * src,
    unsigned int * src_len,
    unsigned char * dest,
    unsigned int * dest_len,
    uint64_t * crc )
```

6.1.2.14 qzDecompressCrc64Ext()

```
int qzDecompressCrc64Ext (
    QzSession_T * sess,
    const unsigned char * src,
    unsigned int * src_len,
    unsigned char * dest,
    unsigned int * dest_len,
    uint64_t * crc,
    uint64_t * ext_rc )
```

6.1.2.15 qzGetDeflateEndOfStream()

```
int qzGetDeflateEndOfStream (
    QzSession_T * sess,
    unsigned char * endofstream )
```

return end of stream.

6.1.2.16 qzMaxCompressedLength()

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

6.1.2.17 qzSetDefaultsDeflate()

```
int qzSetDefaultsDeflate (
    QzSessionParamsDeflate_T * defaults )
```

6.1.2.18 qzSetDefaultsLZ4()

```
int qzSetDefaultsLZ4 (
    QzSessionParamsLZ4_T * defaults )
```

6.1.2.19 qzSetDefaultsLZ4Block()

```
int qzSetDefaultsLZ4Block (
    QzSessionParamsLZ4_T * defaults )
```

6.1.2.20 qzSetDefaultsLZ4S()

```
int qzSetDefaultsLZ4S (
    QzSessionParamsLZ4S_T * defaults )
```

6.1.2.21 qzSetDefaultsZstd()

```
int qzSetDefaultsZstd (
    QzSessionParamsZstd_T * defaults )
```

6.1.2.22 qzSetDefaultsDeflateExt()

```
int qzSetDefaultsDeflateExt (
    QzSessionParamsDeflateExt_T * defaults )
```

6.1.2.23 qzGetDefaultsDeflate()

```
int qzGetDefaultsDeflate (
    QzSessionParamsDeflate_T * defaults )
```

6.1.2.24 qzGetDefaultsLZ4()

```
int qzGetDefaultsLZ4 (
    QzSessionParamsLZ4_T * defaults )
```

6.1.2.25 qzGetDefaultsLZ4Block()

```
int qzGetDefaultsLZ4Block (
    QzSessionParamsLZ4_T * defaults )
```

6.1.2.26 qzGetDefaultsLZ4S()

```
int qzGetDefaultsLZ4S (
    QzSessionParamsLZ4S_T * defaults )
```

6.1.2.27 qzGetDefaultsZstd()

```
int qzGetDefaultsZstd (
    QzSessionParamsZstd_T * defaults )
```

6.1.2.28 qzGetDefaultsDeflateExt()

```
int qzGetDefaultsDeflateExt (
    QzSessionParamsDeflateExt_T * defaults )
```

6.2 qatzip.h

[Go to the documentation of this file.](#)

```
00001 /*****
00002  *
00003  *   BSD LICENSE
00004  *
00005  *   Copyright(c) 2007-2026 Intel Corporation. All rights reserved.
00006  *   All rights reserved.
00007  *
00008  *   Redistribution and use in source and binary forms, with or without
00009  *   modification, are permitted provided that the following conditions
00010  *   are met:
00011  *
00012  *       * Redistributions of source code must retain the above copyright
00013  *       notice, this list of conditions and the following disclaimer.
00014  *       * Redistributions in binary form must reproduce the above copyright
00015  *       notice, this list of conditions and the following disclaimer in
00016  *       the documentation and/or other materials provided with the
00017  *       distribution.
00018  *       * Neither the name of Intel Corporation nor the names of its
00019  *       contributors may be used to endorse or promote products derived
00020  *       from this software without specific prior written permission.
00021  *
00022  *   THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
00023  *   "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
00024  *   LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR
00025  *   A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
00026  *   OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
00027  *   SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT
00028  *   LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE,
00029  *   DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY
00030  *   THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
00031  *   (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE
00032  *   OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
00033  *
00034  *****/
00035
00050 #ifndef INTEL_QATZIP_H
00051 #define INTEL_QATZIP_H
00052
00053 #ifdef __cplusplus
00054 extern"C" {
00055 #endif
00056
00057 #include <string.h>
00058 #include <stdint.h>
00059
00071 #define QATZIP_API_VERSION_NUM_MAJOR (2)
00072
00083 #define QATZIP_API_VERSION_NUM_MINOR (6)
00084
00085 /* Define a macro as an integer to test */
00086 #define QATZIP_API_VERSION (QATZIP_API_VERSION_NUM_MAJOR * 10000 + \
00087                             QATZIP_API_VERSION_NUM_MINOR * 100)
00088
00095 #if defined QATZIP_LINK_DLL
00096 #define QATZIP_API __declspec(dllimport)
00097 #elif defined QATZIP_BUILD_DLL
00098 #define QATZIP_API __declspec(dllexport)
00099 #else
00100 #define QATZIP_API
00101 #endif
00102
```

```

00183 typedef enum QzHuffmanHdr_E {
00184     QZ_DYNAMIC_HDR = 0,
00186     QZ_STATIC_HDR
00188 } QzHuffmanHdr_T;
00189
00200 typedef enum PinMem_E {
00201     COMMON_MEM = 0,
00203     PINNED_MEM
00205 } PinMem_T;
00206
00218 typedef enum QzDirection_E {
00219     QZ_DIR_COMPRESS = 0,
00221     QZ_DIR_DECOMPRESS,
00223     QZ_DIR_BOTH
00225 } QzDirection_T;
00226
00239 typedef enum QzDataFormat_E {
00240     QZ_DEFLATE_4B = 0,
00242     QZ_DEFLATE_GZIP,
00244     QZ_DEFLATE_GZIP_EXT,
00246     QZ_DEFLATE_RAW,
00248     QZ_FMT_NUM
00249 } QzDataFormat_T;
00250
00261 typedef enum QzPollingMode_E {
00262     QZ_PERIODICAL_POLLING = 0,
00264     QZ_BUSY_POLLING,
00266 } QzPollingMode_T;
00267
00278 typedef enum QzCrcType_E {
00279     QZ_CRC32 = 0,
00281     QZ_ADLER,
00283     QZ_XXHASH,
00285     NONE
00287 } QzCrcType_T;
00288
00299 typedef enum QzSoftwareComponentType_E {
00300     QZ_COMPONENT_FIRMWARE = 0,
00301     QZ_COMPONENT_KERNEL_DRIVER,
00302     QZ_COMPONENT_USER_DRIVER,
00303     QZ_COMPONENT_QATZIP_API,
00304     QZ_COMPONENT_SOFTWARE_PROVIDER
00305 } QzSoftwareComponentType_T;
00306
00317 #define QZ_OK (0)
00319 #define QZ_DUPLICATE (1)
00321 #define QZ_FORCE_SW (2)
00323 #define QZ_PARAMS (-1)
00325 #define QZ_FAIL (-2)
00327 #define QZ_BUF_ERROR (-3)
00329 #define QZ_DATA_ERROR (-4)
00331 #define QZ_TIMEOUT (-5)
00333 #define QZ_INTEG (-100)
00335 #define QZ_NO_HW (11)
00337 #define QZ_NO_MDRV (12)
00339 #define QZ_NO_INST_ATTACH (13)
00341 #define QZ_LOW_MEM (14)
00343 #define QZ_LOW_DEST_MEM (15)
00345 #define QZ_UNSUPPORTED_FMT (16)
00347 #define QZ_NONE (100)
00349 #define QZ_NOSW_NO_HW (-101)
00351 #define QZ_NOSW_NO_MDRV (-102)
00353 #define QZ_NOSW_NO_INST_ATTACH (-103)
00355 #define QZ_NOSW_LOW_MEM (-104)
00357 #define QZ_NO_SW_AVAIL (-105)
00359 #define QZ_NOSW_UNSUPPORTED_FMT (-116)
00361 #define QZ_POST_PROCESS_ERROR (-117)
00363 #define QZ_METADATA_OVERFLOW (-118)
00365 #define QZ_OUT_OF_RANGE (-119)
00367 #define QZ_NOT_SUPPORTED (-200)
00370 #define QZ_MAX_ALGORITHMS ((int)255)
00371 #define QZ_DEFLATE ((unsigned char)8)
00374 #define QZ_LZ4 ((unsigned char)'4')
00375 #define QZ_LZ4_BLOCK ((unsigned char)'B')
00376 #define QZ_LZ4s ((unsigned char)'s')
00377 #define QZ_ZSTD ((unsigned char)'Z')
00378
00379 #ifndef MIN
00380 #define MIN(a,b) (((a)<(b))?(a):(b))
00381 #endif
00382 #ifdef __linux__
00383 #define QZ_MEMCPY(dest, src, dest_sz, src_sz) \
00384     memcpy((void *) (dest), (void *) (src), (size_t)MIN(dest_sz, src_sz))
00385 #endif
00386 #ifdef _WIN64
00387 #define QZ_MEMCPY(dest, src, dest_sz, src_sz) \
00388     memcpy_s((void *) (dest), dest_sz, (void *) (src), MIN(dest_sz, src_sz))

```

```

00389 #endif
00390
00454 typedef int (*qzLZ4SCallbackFn)(void *external, const unsigned char *src,
00455                                unsigned int *src_len, unsigned char *dest,
00456                                unsigned int *dest_len, int *ExtStatus);
00457
00467 typedef struct QzSessionParams_S {
00468     QzHuffmanHdr_T huffman_hdr;
00470     QzDirection_T direction;
00472     QzDataFormat_T data_fmt;
00474     unsigned int comp_lvl;
00476     unsigned char comp_algorithm;
00478     unsigned int max_forks;
00481     unsigned char sw_backup;
00483     unsigned int hw_buff_sz;
00486     unsigned int strm_buff_sz;
00489     unsigned int input_sz_thrshold;
00494     unsigned int req_cnt_thrshold;
00497     unsigned int wait_cnt_thrshold;
00500 #ifdef ERR_INJECTION
00501     void *fbError;
00502     void *fbErrorCurr;
00503     /* Linked list for simulated errors from HW */
00504 #endif
00505 } QzSessionParams_T;
00506
00507 typedef struct QzSessionParamsCommon_S {
00508     QzDirection_T direction;
00510     unsigned int comp_lvl;
00512     unsigned char comp_algorithm;
00514     unsigned int max_forks;
00517     unsigned char sw_backup;
00519     unsigned int hw_buff_sz;
00522     unsigned int strm_buff_sz;
00525     unsigned int input_sz_thrshold;
00530     unsigned int req_cnt_thrshold;
00533     unsigned int wait_cnt_thrshold;
00536     QzPollingMode_T polling_mode;
00538     unsigned int is_sensitive_mode;
00540 #ifdef ERR_INJECTION
00541     void *fbError;
00542     void *fbErrorCurr;
00543     /* Linked list for simulated errors from HW */
00544 #endif
00545 } QzSessionParamsCommon_T;
00546
00547 typedef struct QzSessionParamsDeflate_S {
00548     QzSessionParamsCommon_T common_params;
00549     QzHuffmanHdr_T huffman_hdr;
00551     QzDataFormat_T data_fmt;
00553 } QzSessionParamsDeflate_T;
00554
00555 typedef struct QzSessionParamsLZ4_S {
00556     QzSessionParamsCommon_T common_params;
00557 } QzSessionParamsLZ4_T;
00558
00559 typedef struct QzSessionParamsLZ4S_S {
00560     QzSessionParamsCommon_T common_params;
00561     qzLZ4SCallbackFn qzCallback;
00563     void *qzCallback_external;
00566     unsigned int lz4s_mini_match;
00569 } QzSessionParamsLZ4S_T;
00570
00571 typedef struct QzSessionParamsZstd_S {
00572     QzSessionParamsCommon_T common_params;
00573     unsigned int historyBuffer;
00574     unsigned char blockSize;
00575     unsigned char reserved[7];
00576 } QzSessionParamsZstd_T;
00577
00578 typedef struct QzDictionaryParams_S {
00579     const unsigned char *dictionary;
00580     /*< Pointer to dictionary buffer */
00581     unsigned int dictionary_len;
00582     /*< Length of dictionary buffer (maximum value of 32KB) */
00583     int compressed; /*< non-zero indicates compressed dictionary supplied */
00584     unsigned long *crc64Addr; /*< If not NULL, address to store CRC64 generated
00585                               on the dictionary */
00586     unsigned int dictId; /*< Dictionary ID for ZSTD format (0 if not used) */
00587 } QzDictionaryParams_T;
00588
00589 typedef struct QzSessionParamsDeflateExt_S {
00590     QzSessionParamsDeflate_T deflate_params;
00591     /* stop decompression at end the stream, default is 0. */
00592     unsigned char stop_decompression_stream_end;
00594     unsigned char zlib_format;
00595 } QzSessionParamsDeflateExt_T;

```

```

00596
00597 #define QZ_HUFF_HDR_DEFAULT          QZ_DYNAMIC_HDR
00598 #define QZ_DIRECTION_DEFAULT         QZ_DIR_BOTH
00599 #define QZ_DATA_FORMAT_DEFAULT       QZ_DEFLATE_GZIP_EXT
00600 #define QZ_COMP_LEVEL_DEFAULT        1
00601 #define QZ_COMP_ALGOL_DEFAULT        QZ_DEFLATE
00602 #define QZ_POLL_SLEEP_DEFAULT        10
00603 #define QZ_MAX_FORK_DEFAULT          3
00604 #define QZ_SW_BACKUP_DEFAULT         1
00605 #define QZ_HW_BUFF_SZ                (64*1024)
00606 #define QZ_HW_BUFF_SZ_Gen3           (1*1024*1024)
00607 #define QZ_HW_BUFF_MIN_SZ            (1*1024)
00608 #define QZ_HW_BUFF_MAX_SZ            (512*1024)
00609 #define QZ_HW_BUFF_MAX_SZ_Gen3       (2*1024*1024*1024U)
00610 #define QZ_STRM_BUFF_SZ_DEFAULT      QZ_HW_BUFF_SZ
00611 #define QZ_STRM_BUFF_MIN_SZ          (1*1024)
00612 #define QZ_STRM_BUFF_MAX_SZ          (2*1024*1024 - 5*1024)
00613 #define QZ_COMP_THRESHOLD_DEFAULT    1024
00614 #define QZ_COMP_THRESHOLD_MINIMUM    128
00615 #define QZ_REQ_THRESHOLD_MINIMUM     1
00616 #define QZ_REQ_THRESHOLD_MAXIMUM     NUM_BUFF
00617 #define QZ_REQ_THRESHOLD_DEFAULT     QZ_REQ_THRESHOLD_MAXIMUM
00618 #define QZ_WAIT_CNT_THRESHOLD_DEFAULT 8
00619 #define QZ_DEFLATE_COMP_LVL_MINIMUM   (1)
00620 #define QZ_DEFLATE_COMP_LVL_MAXIMUM   (9)
00621 #define QZ_DEFLATE_COMP_LVL_MAXIMUM_Gen3 (12)
00622 #define QZ_LZS_COMP_LVL_MINIMUM       (1)
00623 #define QZ_LZS_COMP_LVL_MAXIMUM       (12)
00624 #define QZ_AUTO_SELECT_NUMA_NODE      (-1)
00625 #define QZ_ZSTD_COMP_LVL_MINIMUM      (1)
00626 #define QZ_ZSTD_COMP_LVL_MAXIMUM      (12)
00627 #define QZ_ZSTD_HIST_BUFF_DEFAULT      (64*1024)
00628 #define QZ_ZSTD_BLOCKSIZE_DEFAULT     (128)
00629 #define QZ_ZSTD_HW_BUFF_SZ            (128*1024)
00630
00646 #define QZ_SW_BACKUP_BIT_POSITION     (0)
00647 #define QZ_SW_FORCESW_BIT_POSITION    (1)
00648
00649 #define QZ_ENABLE_SOFTWARE_BACKUP(_BackupVariable) \
00650     (_BackupVariable != (1 < QZ_SW_BACKUP_BIT_POSITION))
00652 #define QZ_ENABLE_SOFTWARE_ONLY_EXECUTION(_BackupVariable) \
00653     (_BackupVariable != (1 < QZ_SW_FORCESW_BIT_POSITION))
00656 #define QZ_DISABLE_SOFTWARE_BACKUP(_BackupVariable) \
00657     (_BackupVariable &= ~(1 < QZ_SW_BACKUP_BIT_POSITION))
00659 #define QZ_DISABLE_SOFTWARE_ONLY_EXECUTION(_BackupVariable) \
00660     (_BackupVariable &= ~(1 < QZ_SW_FORCESW_BIT_POSITION))
00680 #define QZ_SW_EXECUTION_BIT           (4)
00681 #define QZ_SW_EXECUTION_MASK          (1 < QZ_SW_EXECUTION_BIT)
00682 #define QZ_SW_EXECUTION(ret, ext_rc) \
00683     (!ret && (ext_rc & QZ_SW_EXECUTION_MASK))
00684
00685 #define QZ_TIMEOUT_BIT                (8)
00686 #define QZ_TIMEOUT_MASK               (1 < QZ_TIMEOUT_BIT)
00687 #define QZ_HW_TIMEOUT(ret, ext_rc) \
00688     (!ret && (ext_rc & QZ_TIMEOUT_MASK))
00689
00690 #define QZ_POST_PROCESS_FAIL_BIT       (10)
00691 #define QZ_POST_PROCESS_FAIL_MASK     (1 < QZ_POST_PROCESS_FAIL_BIT)
00692 #define QZ_POST_PROCESS_FAIL(ret, ext_rc) \
00693     (ret && (ext_rc & QZ_POST_PROCESS_FAIL_MASK))
00694
00705 typedef struct QzSession_S {
00706     signed long int hw_session_stat;
00708     int thd_sess_stat;
00710     void *internal;
00712     unsigned long total_in;
00714     unsigned long total_out;
00716 } QzSession_T;
00717
00728 typedef struct QzStatus_S {
00729     unsigned short int qat_hw_count;
00731     unsigned char qat_service_init;
00733     unsigned char qat_mem_drvr;
00737     unsigned char qat_instance_attach;
00739     unsigned long int memory_allocated;
00741     unsigned char using_huge_pages;
00743     signed long int hw_session_status;
00745     unsigned char algo_sw[QZ_MAX_ALGORITHMS];
00747     unsigned char algo_hw[QZ_MAX_ALGORITHMS];
00749 } QzStatus_T;
00750
00761 #define QZ_MAX_STRING_LENGTH 64
00762 typedef struct QzSoftwareVersionInfo_S {
00763     QzSoftwareComponentType_T component_type;
00764     unsigned char component_name[QZ_MAX_STRING_LENGTH];
00765     unsigned int major_version;
00766     unsigned int minor_version;

```

```

00767     unsigned int patch_version;
00768     unsigned int build_number;
00769     unsigned char reserved[52];
00770 } QzSoftwareVersionInfo_T;
00771
00782 typedef struct QzCrc64Config_S {
00783     uint64_t polynomial;
00785     uint64_t initial_value;
00787     uint32_t reflect_in;
00789     uint32_t reflect_out;
00791     uint64_t xor_out;
00793 } QzCrc64Config_T;
00794
00805 typedef struct QzCrc32Config_S {
00806     uint32_t polynomial;
00808     uint32_t initial_value;
00810     uint32_t reflect_in;
00812     uint32_t reflect_out;
00814     uint32_t xor_out;
00816 } QzCrc32Config_T;
00817
00834 #define QZ_INPUT_CRC_VALID_BIT                (4)
00835 #define QZ_INPUT_CRC_VALID_MASK                (1 << QZ_INPUT_CRC_VALID_BIT)
00836 #define QZ_OUTPUT_CRC_VALID_BIT                (8)
00837 #define QZ_OUTPUT_CRC_VALID_MASK              (1 << QZ_OUTPUT_CRC_VALID_BIT)
00838 #define QZ_CRC32_VALID_BIT                    (12)
00839 #define QZ_CRC32_VALID_MASK                  (1 << QZ_CRC32_VALID_BIT)
00840 #define QZ_CRC64_VALID_BIT                    (16)
00841 #define QZ_CRC64_VALID_MASK                  (1 << QZ_CRC64_VALID_BIT)
00842
00843 #define QZ_CRC32_VALID(valid_flags) \
00844     ((valid_flags & QZ_CRC32_VALID_MASK) && !(valid_flags & QZ_CRC64_VALID_MASK))
00845
00846 #define QZ_CRC64_VALID(valid_flags) \
00847     ((valid_flags & QZ_CRC64_VALID_MASK) && !(valid_flags & QZ_CRC32_VALID_MASK))
00848
00849 #define QZ_INPUT_CRC_VALID(valid_flags) \
00850     (valid_flags & QZ_INPUT_CRC_VALID_MASK)
00851
00852 #define QZ_OUTPUT_CRC_VALID(valid_flags) \
00853     (valid_flags & QZ_OUTPUT_CRC_VALID_BIT)
00854
00865 typedef struct QzCrcResult_S {
00866     int status;
00868     uint32_t valid_flags;
00870     union {
00871         uint32_t *crc_32;
00872         uint64_t *crc_64;
00873     } in_crc;
00875     union {
00876         uint32_t *crc_32;
00877         uint64_t *crc_64;
00878     } out_crc;
00880 } QzCrcResult_T;
00881
00894 typedef struct QzResult_S {
00895     int status;
00897     void *cb_tag;
00902     unsigned int src_len;
00904     unsigned int dest_len;
00907     uint64_t ext_rc;
00909     QzCrcResult_T *crc;
00911     void *extension_result;
00914 } QzResult_T;
00915
00951 typedef int (*qzAsyncCallbackFn)(QzResult_T *res);
00952
00962 typedef void *QzMetadataBlob_T;
00963
00973 typedef enum QzLogLevel_E {
00974     LOG_NONE = 0,
00975     LOG_FATAL,
00976     LOG_ERROR,
00977     LOG_WARNING,
00978     LOG_INFO,
00979     LOG_DEBUG1,
00980     LOG_DEBUG2,
00981     LOG_DEBUG3
00982 } QzLogLevel_T;
00983
01019 QATZIP_API QzLogLevel_T qzSetLogLevel(QzLogLevel_T level);
01020
01097 QATZIP_API int qzInit(QzSession_T *sess, unsigned char sw_backup);
01098
01168 QATZIP_API int qzSetupSession(QzSession_T *sess, QzSessionParams_T *params);
01169
01170 QATZIP_API int qzSetupSessionDeflate(QzSession_T *sess,

```



```

01171                                     QzSessionParamsDeflate_T *params);
01172
01173 QATZIP_API int qzSetupSessionLZ4(QzSession_T *sess,
01174                                   QzSessionParamsLZ4_T *params);
01175
01176 QATZIP_API int qzSetupSessionLZ4Block(QzSession_T *sess,
01177                                       QzSessionParamsLZ4_T *params);
01178
01179 QATZIP_API int qzSetupSessionLZ4S(QzSession_T *sess,
01180                                   QzSessionParamsLZ4S_T *params);
01181
01182 QATZIP_API int qzSetupSessionZstd(QzSession_T *sess,
01183                                   QzSessionParamsZstd_T *params);
01184
01185 QATZIP_API int qzSetupSessionDeflateExt(QzSession_T *sess,
01186                                         QzSessionParamsDeflateExt_T *params);
01257 QATZIP_API int qzCompress(QzSession_T *sess, const unsigned char *src,
01258                          unsigned int *src_len, unsigned char *dest,
01259                          unsigned int *dest_len, unsigned int last);
01260
01261 QATZIP_API int qzCompressExt(QzSession_T *sess, const unsigned char *src,
01262                              unsigned int *src_len, unsigned char *dest,
01263                              unsigned int *dest_len, unsigned int last,
01264                              uint64_t *ext_rc);
01265
01337 QATZIP_API int qzCompressCrc(QzSession_T *sess,
01338                              const unsigned char *src,
01339                              unsigned int *src_len,
01340                              unsigned char *dest,
01341                              unsigned int *dest_len,
01342                              unsigned int last,
01343                              unsigned long *crc);
01344
01345 QATZIP_API int qzCompressCrcExt(QzSession_T *sess,
01346                                 const unsigned char *src,
01347                                 unsigned int *src_len,
01348                                 unsigned char *dest,
01349                                 unsigned int *dest_len,
01350                                 unsigned int last,
01351                                 unsigned long *crc,
01352                                 uint64_t *ext_rc);
01353
01354 QATZIP_API int qzCompressCrc64(QzSession_T *sess,
01355                                const unsigned char *src,
01356                                unsigned int *src_len,
01357                                unsigned char *dest,
01358                                unsigned int *dest_len,
01359                                unsigned int last,
01360                                uint64_t *crc);
01361
01362 QATZIP_API int qzCompressCrc64Ext(QzSession_T *sess,
01363                                   const unsigned char *src,
01364                                   unsigned int *src_len,
01365                                   unsigned char *dest,
01366                                   unsigned int *dest_len,
01367                                   unsigned int last,
01368                                   uint64_t *crc,
01369                                   uint64_t *ext_rc);
01370
01474 QATZIP_API int qzCompressWithMetadataExt(QzSession_T *sess,
01475                                           const unsigned char *src,
01476                                           unsigned int *src_len,
01477                                           unsigned char *dest,
01478                                           unsigned int *dest_len,
01479                                           unsigned int last,
01480                                           uint64_t *ext_rc,
01481                                           QzMetadataBlob_T metadata,
01482                                           uint32_t hw_buff_sz_override,
01483                                           uint32_t comp_thrshold);
01484
01537 QATZIP_API int qzCompressWithDictionary(QzSession_T *sess,
01538                                          const unsigned char *src,
01539                                          unsigned char *dest,
01540                                          QzDictionaryParams_T *dictionary,
01541                                          qzAsyncCallbackFn callback,
01542                                          QzResult_T *qzResults);
01543
01544 QATZIP_API int qzCompress2(QzSession_T *sess, const unsigned char *src,
01545                            unsigned char *dest, qzAsyncCallbackFn callback,
01546                            QzResult_T *qzResults);
01547
01603 QATZIP_API int qzDecompress(QzSession_T *sess, const unsigned char *src,
01604                             unsigned int *src_len, unsigned char *dest,
01605                             unsigned int *dest_len);
01606
01607 QATZIP_API int qzDecompressExt(QzSession_T *sess, const unsigned char *src,
01608                                unsigned int *src_len, unsigned char *dest,

```

```

01609                                     unsigned int *dest_len, uint64_t *ext_rc);
01610
01671 QATZIP_API int qzDecompressCrc(QzSession_T *sess,
01672                               const unsigned char *src,
01673                               unsigned int *src_len,
01674                               unsigned char *dest,
01675                               unsigned int *dest_len,
01676                               unsigned long *crc);
01677
01678 QATZIP_API int qzDecompressCrcExt(QzSession_T *sess,
01679                                  const unsigned char *src,
01680                                  unsigned int *src_len,
01681                                  unsigned char *dest,
01682                                  unsigned int *dest_len,
01683                                  unsigned long *crc,
01684                                  uint64_t *ext_rc);
01685
01686 QATZIP_API int qzDecompressCrc64(QzSession_T *sess,
01687                                  const unsigned char *src,
01688                                  unsigned int *src_len,
01689                                  unsigned char *dest,
01690                                  unsigned int *dest_len,
01691                                  uint64_t *crc);
01692
01693 QATZIP_API int qzDecompressCrc64Ext(QzSession_T *sess,
01694                                     const unsigned char *src,
01695                                     unsigned int *src_len,
01696                                     unsigned char *dest,
01697                                     unsigned int *dest_len,
01698                                     uint64_t *crc,
01699                                     uint64_t *ext_rc);
01700
01752 QATZIP_API int qzDecompressWithDictionary(QzSession_T *sess,
01753                                           const unsigned char *src,
01754                                           unsigned char *dest,
01755                                           QzDictionaryParams_T *dictionary,
01756                                           qzAsyncCallbackFn callback,
01757                                           QzResult_T *qzResults);
01758
01839 QATZIP_API int qzDecompressWithMetadataExt(QzSession_T *sess,
01840                                             const unsigned char *src,
01841                                             unsigned int *src_len,
01842                                             unsigned char *dest,
01843                                             unsigned int *dest_len,
01844                                             uint64_t *ext_rc,
01845                                             QzMetadataBlob_T metadata,
01846                                             uint32_t hw_buff_sz_override);
01847
01899 QATZIP_API int qzDecompress2(QzSession_T *sess, const unsigned char *src,
01900                               unsigned char *dest, qzAsyncCallbackFn callback,
01901                               QzResult_T *qzResults);
01902
01944 QATZIP_API int qzTeardownSession(QzSession_T *sess);
01945
01985 QATZIP_API int qzClose(QzSession_T *sess);
01986
02037 QATZIP_API int qzCompress2(QzSession_T *sess, const unsigned char *src,
02038                           unsigned char *dest, qzAsyncCallbackFn callback,
02039                           QzResult_T *qzResults);
02040
02139 QATZIP_API int qzGetStatus(QzSession_T *sess, QzStatus_T *status);
02140
02145 QATZIP_API int qzGetDeflateEndOfStream(QzSession_T *sess,
02146                                         unsigned char *endofstream);
02147
02189 #define QZ_SKID_PAD_SZ 48
02190 #define QZ_COMPRESSED_SZ_OF_EMPTY_FILE 34
02191 QATZIP_API
02192 unsigned int qzMaxCompressedLength(unsigned int src_sz, QzSession_T *sess);
02193
02232 QATZIP_API int qzSetDefaults(QzSessionParams_T *defaults);
02233
02234 QATZIP_API int qzSetDefaultsDeflate(QzSessionParamsDeflate_T *defaults);
02235
02236 QATZIP_API int qzSetDefaultsLZ4(QzSessionParamsLZ4_T *defaults);
02237
02238 QATZIP_API int qzSetDefaultsLZ4Block(QzSessionParamsLZ4_T *defaults);
02239
02240 QATZIP_API int qzSetDefaultsLZ4S(QzSessionParamsLZ4S_T *defaults);
02241
02242 QATZIP_API int qzSetDefaultsZstd(QzSessionParamsZstd_T *defaults);
02243
02244 QATZIP_API int qzSetDefaultsDeflateExt(QzSessionParamsDeflateExt_T *defaults);
02245
02283 QATZIP_API int qzGetDefaults(QzSessionParams_T *defaults);
02284
02285 QATZIP_API int qzGetDefaultsDeflate(QzSessionParamsDeflate_T *defaults);

```

```

02286
02287 QATZIP_API int qzGetDefaultsLZ4(QzSessionParamsLZ4_T *defaults);
02288
02289 QATZIP_API int qzGetDefaultsLZ4Block(QzSessionParamsLZ4_T *defaults);
02290
02291 QATZIP_API int qzGetDefaultsLZ4S(QzSessionParamsLZ4S_T *defaults);
02292
02293 QATZIP_API int qzGetDefaultsZstd(QzSessionParamsZstd_T *defaults);
02294
02295 QATZIP_API int qzGetDefaultsDeflateExt(QzSessionParamsDeflateExt_T *defaults);
02296
02348 QATZIP_API int qzDecompress2(QzSession_T *sess, const unsigned char *src,
02349                               unsigned char *dest, qzAsyncCallbackFn callback,
02350                               QzResult_T *qzResults);
02392 QATZIP_API void *qzMalloc(size_t sz, int numa, int force_pinned);
02393
02439 QATZIP_API int qzAllocateMetadata(QzMetadataBlob_T *metadata,
02440                                   size_t data_size,
02441                                   uint32_t hw_buff_sz);
02442
02477 QATZIP_API void qzFree(void *m);
02478
02517 QATZIP_API int qzFreeMetadata(QzMetadataBlob_T metadata);
02518
02557 QATZIP_API int qzMemFindAddr(unsigned char *a);
02558
02568 typedef struct QzStream_S {
02569     unsigned int in_sz;
02571     unsigned int out_sz;
02573     unsigned char *in ;
02575     unsigned char *out ;
02577     unsigned int pending_in;
02579     unsigned int pending_out;
02581     QzCrcType_T crc_type;
02583     unsigned int crc_32;
02585     unsigned long long reserved;
02587     void *opaque;
02589 } QzStream_T;
02590
02666 QATZIP_API
02667 int qzCompressStream(QzSession_T *sess, QzStream_T *strm, unsigned int last);
02668
02743 QATZIP_API
02744 int qzDecompressStream(QzSession_T *sess, QzStream_T *strm, unsigned int last);
02745
02786 QATZIP_API int qzEndStream(QzSession_T *sess, QzStream_T *strm);
02787
02838 QATZIP_API
02839 int qzGetSoftwareComponentVersionList(QzSoftwareVersionInfo_T *api_info,
02840                                       unsigned int *num_elem);
02841
02888 QATZIP_API int qzGetSoftwareComponentCount(unsigned int *num_elem);
02889
02932 QATZIP_API int qzGetSessionCrc64Config(QzSession_T *sess,
02933                                         QzCrc64Config_T *crc64_config);
02934
02977 QATZIP_API int qzGetSessionCrc32Config(QzSession_T *sess,
02978                                         QzCrc32Config_T *crc32_config);
02979
03024 QATZIP_API int qzSetSessionCrc64Config(QzSession_T *sess,
03025                                         QzCrc64Config_T *crc64_config);
03026
03071 QATZIP_API int qzSetSessionCrc32Config(QzSession_T *sess,
03072                                         QzCrc32Config_T *crc32_config);
03073
03137 QATZIP_API int qzMetadataBlockRead(uint32_t block_num,
03138                                     QzMetadataBlob_T metadata,
03139                                     uint32_t *block_offset,
03140                                     uint32_t *block_size,
03141                                     uint32_t *block_flags,
03142                                     uint32_t *block_hash);
03143
03206 QATZIP_API int qzMetadataBlockWrite(uint32_t block_num,
03207                                     QzMetadataBlob_T metadata,
03208                                     uint32_t *block_offset,
03209                                     uint32_t *block_size,
03210                                     uint32_t *block_flags,
03211                                     uint32_t *block_hash);
03212
03258 QATZIP_API int qzMetadataBlockGetCrc64(uint32_t block_num,
03259                                         QzMetadataBlob_T metadata,
03260                                         uint64_t *input_crc,
03261                                         uint64_t *output_crc);
03262
03308 QATZIP_API int qzMetadataBlockGetCrc32(uint32_t block_num,
03309                                         QzMetadataBlob_T metadata,
03310                                         uint32_t *input_crc,

```

```
03311                                uint32_t *output_crc);
03312
03313 #ifdef __cplusplus
03314 }
03315 #endif
03316
03317 #endif /* INTEL_QATZIP_H */
```

Index

algo_hw
 QzStatus_T, 90
algo_sw
 QzStatus_T, 90

blockSize
 QzSessionParamsZstd_T, 87
build_number
 QzSoftwareVersionInfo_T, 88

cb_tag
 QzResult_T, 78
COMMON_MEM
 Data Compression API, 17
common_params
 QzSessionParamsDeflate_T, 85
 QzSessionParamsLZ4_T, 86
 QzSessionParamsLZ4S_T, 86
 QzSessionParamsZstd_T, 87
comp_algorithm
 QzSessionParams_T, 81
 QzSessionParamsCommon_T, 83
comp_lvl
 QzSessionParams_T, 81
 QzSessionParamsCommon_T, 83
component_name
 QzSoftwareVersionInfo_T, 88
component_type
 QzSoftwareVersionInfo_T, 88
compressed
 QzDictionaryParams_T, 77
crc
 QzResult_T, 79
crc64Addr
 QzDictionaryParams_T, 77
crc_32
 QzCrcResult_T, 76
 QzStream_T, 92
crc_64
 QzCrcResult_T, 76
crc_type
 QzStream_T, 92

Data Compression API, 7
 COMMON_MEM, 17
 LOG_DEBUG1, 19
 LOG_DEBUG2, 19
 LOG_DEBUG3, 19
 LOG_ERROR, 19
 LOG_FATAL, 19
 LOG_INFO, 19
 LOG_NONE, 19
 LOG_WARNING, 19
 NONE, 18
 PinMem_T, 16
 PINNED_MEM, 17
 QATZIP_API_VERSION_NUM_MAJOR, 9
 QATZIP_API_VERSION_NUM_MINOR, 9
 QZ_ADLER, 18
 QZ_BUSY_POLLING, 18
 QZ_COMPONENT_FIRMWARE, 19
 QZ_COMPONENT_KERNEL_DRIVER, 19
 QZ_COMPONENT_QATZIP_API, 19
 QZ_COMPONENT_SOFTWARE_PROVIDER, 19
 QZ_COMPONENT_USER_DRIVER, 19
 QZ_CRC32, 18
 QZ_DEFLATE_4B, 17
 QZ_DEFLATE_GZIP, 17
 QZ_DEFLATE_GZIP_EXT, 17
 QZ_DEFLATE_RAW, 17
 QZ_DIR_BOTH, 17
 QZ_DIR_COMPRESS, 17
 QZ_DIR_DECOMPRESS, 17
 QZ_DYNAMIC_HDR, 16
 QZ_FMT_NUM, 17
 QZ_INPUT_CRC_VALID_BIT, 11
 QZ_MAX_STRING_LENGTH, 10
 QZ_OK, 9
 QZ_PERIODICAL_POLLING, 18
 QZ_SKID_PAD_SZ, 11
 QZ_STATIC_HDR, 16
 QZ_SW_BACKUP_BIT_POSITION, 10
 QZ_SW_EXECUTION_BIT, 10
 QZ_XXHASH, 18
 qzAllocateMetadata, 48
 qzAsyncCallbackFn, 14
 qzClose, 40
 qzCompress, 24
 qzCompress2, 41
 qzCompressCrc, 25
 qzCompressStream, 53
 qzCompressWithDictionary, 29
 qzCompressWithMetadataExt, 27
 QzCrcType_T, 18
 QzDataFormat_T, 17
 qzDecompress, 30
 qzDecompress2, 37
 qzDecompressCrc, 32
 qzDecompressStream, 54

- qzDecompressWithDictionary, 34
- qzDecompressWithMetadataExt, 35
- QzDirection_T, 17
- qzEndStream, 56
- qzFree, 49
- qzFreeMetadata, 51
- qzGetDefaults, 46
- qzGetSessionCrc32Config, 61
- qzGetSessionCrc64Config, 60
- qzGetSoftwareComponentCount, 59
- qzGetSoftwareComponentVersionList, 57
- qzGetStatus, 42
- QzHuffmanHdr_T, 15
- qzInit, 20
- QzLogLevel_T, 19
- qzLZ4SCallbackFn, 12
- qzMalloc, 47
- qzMemFindAddr, 52
- QzMetadataBlob_T, 15
- qzMetadataBlockGetCrc32, 70
- qzMetadataBlockGetCrc64, 69
- qzMetadataBlockRead, 65
- qzMetadataBlockWrite, 67
- QzPollingMode_T, 18
- qzSetDefaults, 44
- qzSetLogLevel, 19
- qzSetSessionCrc32Config, 64
- qzSetSessionCrc64Config, 63
- qzSetupSession, 22
- QzSoftwareComponentType_T, 18
- qzTeardownSession, 38
- data_fmt
 - QzSessionParams_T, 81
 - QzSessionParamsDeflate_T, 85
- deflate_params
 - QzSessionParamsDeflateExt_T, 85
- dest_len
 - QzResult_T, 78
- dictId
 - QzDictionaryParams_T, 77
- dictionary
 - QzDictionaryParams_T, 77
- dictionary_len
 - QzDictionaryParams_T, 77
- direction
 - QzSessionParams_T, 81
 - QzSessionParamsCommon_T, 83
- ext_rc
 - QzResult_T, 78
- extension_result
 - QzResult_T, 79
- historyBuffer
 - QzSessionParamsZstd_T, 87
- huffman_hdr
 - QzSessionParams_T, 81
 - QzSessionParamsDeflate_T, 85
- hw_buff_sz
 - QzSessionParams_T, 82
 - QzSessionParamsCommon_T, 83
- hw_session_stat
 - QzSession_T, 79
- hw_session_status
 - QzStatus_T, 90
- in
 - QzStream_T, 91
- in_crc
 - QzCrcResult_T, 76
- in_sz
 - QzStream_T, 91
- initial_value
 - QzCrc32Config_T, 73
 - QzCrc64Config_T, 75
- input_sz_threshold
 - QzSessionParams_T, 82
 - QzSessionParamsCommon_T, 84
- internal
 - QzSession_T, 80
- is_sensitive_mode
 - QzSessionParamsCommon_T, 84
- LOG_DEBUG1
 - Data Compression API, 19
- LOG_DEBUG2
 - Data Compression API, 19
- LOG_DEBUG3
 - Data Compression API, 19
- LOG_ERROR
 - Data Compression API, 19
- LOG_FATAL
 - Data Compression API, 19
- LOG_INFO
 - Data Compression API, 19
- LOG_NONE
 - Data Compression API, 19
- LOG_WARNING
 - Data Compression API, 19
- lz4s_mini_match
 - QzSessionParamsLZ4S_T, 87
- major_version
 - QzSoftwareVersionInfo_T, 88
- max_forks
 - QzSessionParams_T, 81
 - QzSessionParamsCommon_T, 83
- memory_allocated
 - QzStatus_T, 90
- MIN
 - qatzip.h, 101
- minor_version
 - QzSoftwareVersionInfo_T, 88
- NONE
 - Data Compression API, 18
- opaque

- QzStream_T, 92
- out
 - QzStream_T, 91
- out_crc
 - QzCrcResult_T, 76
- out_sz
 - QzStream_T, 91
- patch_version
 - QzSoftwareVersionInfo_T, 88
- pending_in
 - QzStream_T, 91
- pending_out
 - QzStream_T, 92
- PinMem_T
 - Data Compression API, 16
- PINNED_MEM
 - Data Compression API, 17
- polling_mode
 - QzSessionParamsCommon_T, 84
- polynomial
 - QzCrc32Config_T, 73
 - QzCrc64Config_T, 75
- qat_hw_count
 - QzStatus_T, 89
- qat_instance_attach
 - QzStatus_T, 89
- qat_mem_drvr
 - QzStatus_T, 89
- qat_service_init
 - QzStatus_T, 89
- qatzip.h, 93, 112
 - MIN, 101
 - QATZIP_API, 97
 - QATZIP_API_VERSION, 97
 - QZ_AUTO_SELECT_NUMA_NODE, 104
 - QZ_BUF_ERROR, 98
 - QZ_COMP_ALGOL_DEFAULT, 102
 - QZ_COMP_LEVEL_DEFAULT, 102
 - QZ_COMP_THRESHOLD_DEFAULT, 103
 - QZ_COMP_THRESHOLD_MINIMUM, 103
 - QZ_COMPRESSED_SZ_OF_EMPTY_FILE, 107
 - QZ_CRC32_VALID, 107
 - QZ_CRC32_VALID_BIT, 107
 - QZ_CRC32_VALID_MASK, 107
 - QZ_CRC64_VALID, 107
 - QZ_CRC64_VALID_BIT, 107
 - QZ_CRC64_VALID_MASK, 107
 - QZ_DATA_ERROR, 98
 - QZ_DATA_FORMAT_DEFAULT, 102
 - QZ_DEFLATE, 101
 - QZ_DEFLATE_COMP_LVL_MAXIMUM, 104
 - QZ_DEFLATE_COMP_LVL_MAXIMUM_Gen3, 104
 - QZ_DEFLATE_COMP_LVL_MINIMUM, 104
 - QZ_DIRECTION_DEFAULT, 102
 - QZ_DISABLE_SOFTWARE_BACKUP, 105
 - QZ_DISABLE_SOFTWARE_ONLY_EXECUTION, 105
 - QZ_DUPLICATE, 98
 - QZ_ENABLE_SOFTWARE_BACKUP, 105
 - QZ_ENABLE_SOFTWARE_ONLY_EXECUTION, 105
 - QZ_FAIL, 98
 - QZ_FORCE_SW, 98
 - QZ_HUFF_HDR_DEFAULT, 102
 - QZ_HW_BUFF_MAX_SZ, 103
 - QZ_HW_BUFF_MAX_SZ_Gen3, 103
 - QZ_HW_BUFF_MIN_SZ, 103
 - QZ_HW_BUFF_SZ, 102
 - QZ_HW_BUFF_SZ_Gen3, 102
 - QZ_HW_TIMEOUT, 106
 - QZ_INPUT_CRC_VALID, 107
 - QZ_INPUT_CRC_VALID_MASK, 106
 - QZ_INTEG, 99
 - QZ_LOW_DEST_MEM, 99
 - QZ_LOW_MEM, 99
 - QZ_LZ4, 101
 - QZ_LZ4_BLOCK, 101
 - QZ_LZ4s, 101
 - QZ_LZS_COMP_LVL_MAXIMUM, 104
 - QZ_LZS_COMP_LVL_MINIMUM, 104
 - QZ_MAX_ALGORITHMS, 101
 - QZ_MAX_FORK_DEFAULT, 102
 - QZ_METADATA_OVERFLOW, 100
 - QZ_NO_HW, 99
 - QZ_NO_INST_ATTACH, 99
 - QZ_NO_MDRV, 99
 - QZ_NO_SW_AVAIL, 100
 - QZ_NONE, 99
 - QZ_NOSW_LOW_MEM, 100
 - QZ_NOSW_NO_HW, 100
 - QZ_NOSW_NO_INST_ATTACH, 100
 - QZ_NOSW_NO_MDRV, 100
 - QZ_NOSW_UNSUPPORTED_FMT, 100
 - QZ_NOT_SUPPORTED, 101
 - QZ_OUT_OF_RANGE, 101
 - QZ_OUTPUT_CRC_VALID, 107
 - QZ_OUTPUT_CRC_VALID_BIT, 106
 - QZ_OUTPUT_CRC_VALID_MASK, 107
 - QZ_PARAMS, 98
 - QZ_POLL_SLEEP_DEFAULT, 102
 - QZ_POST_PROCESS_ERROR, 100
 - QZ_POST_PROCESS_FAIL, 106
 - QZ_POST_PROCESS_FAIL_BIT, 106
 - QZ_POST_PROCESS_FAIL_MASK, 106
 - QZ_REQ_THRESHOLD_DEFAULT, 104
 - QZ_REQ_THRESHOLD_MAXIMUM, 103
 - QZ_REQ_THRESHOLD_MINIMUM, 103
 - QZ_STRM_BUFF_MAX_SZ, 103
 - QZ_STRM_BUFF_MIN_SZ, 103
 - QZ_STRM_BUFF_SZ_DEFAULT, 103
 - QZ_SW_BACKUP_DEFAULT, 102
 - QZ_SW_EXECUTION, 106
 - QZ_SW_EXECUTION_MASK, 105

- QZ_SW_FORCESW_BIT_POSITION, 105
- QZ_TIMEOUT, 98
- QZ_TIMEOUT_BIT, 106
- QZ_TIMEOUT_MASK, 106
- QZ_UNSUPPORTED_FMT, 99
- QZ_WAIT_CNT_THRESHOLD_DEFAULT, 104
- QZ_ZSTD, 101
- QZ_ZSTD_BLOCKSIZE_DEFAULT, 105
- QZ_ZSTD_COMP_LVL_MAXIMUM, 104
- QZ_ZSTD_COMP_LVL_MINIMUM, 104
- QZ_ZSTD_HIST_BUFF_DEFAULT, 105
- QZ_ZSTD_HW_BUFF_SZ, 105
- qzCompressCrc64, 109
- qzCompressCrc64Ext, 109
- qzCompressCrcExt, 109
- qzCompressExt, 108
- qzDecompressCrc64, 110
- qzDecompressCrc64Ext, 110
- qzDecompressCrcExt, 109
- qzDecompressExt, 109
- qzGetDefaultsDeflate, 111
- qzGetDefaultsDeflateExt, 112
- qzGetDefaultsLZ4, 111
- qzGetDefaultsLZ4Block, 111
- qzGetDefaultsLZ4S, 111
- qzGetDefaultsZstd, 111
- qzGetDeflateEndOfStream, 110
- qzMaxCompressedLength, 110
- qzSetDefaultsDeflate, 110
- qzSetDefaultsDeflateExt, 111
- qzSetDefaultsLZ4, 110
- qzSetDefaultsLZ4Block, 111
- qzSetDefaultsLZ4S, 111
- qzSetDefaultsZstd, 111
- qzSetupSessionDeflate, 108
- qzSetupSessionDeflateExt, 108
- qzSetupSessionLZ4, 108
- qzSetupSessionLZ4Block, 108
- qzSetupSessionLZ4S, 108
- qzSetupSessionZstd, 108
- QATZIP_API
 - qatzip.h, 97
- QATZIP_API_VERSION
 - qatzip.h, 97
- QATZIP_API_VERSION_NUM_MAJOR
 - Data Compression API, 9
- QATZIP_API_VERSION_NUM_MINOR
 - Data Compression API, 9
- QZ_ADLER
 - Data Compression API, 18
- QZ_AUTO_SELECT_NUMA_NODE
 - qatzip.h, 104
- QZ_BUF_ERROR
 - qatzip.h, 98
- QZ_BUSY_POLLING
 - Data Compression API, 18
- QZ_COMP_ALGOL_DEFAULT
 - qatzip.h, 102
- QZ_COMP_LEVEL_DEFAULT
 - qatzip.h, 102
- QZ_COMP_THRESHOLD_DEFAULT
 - qatzip.h, 103
- QZ_COMP_THRESHOLD_MINIMUM
 - qatzip.h, 103
- QZ_COMPONENT_FIRMWARE
 - Data Compression API, 19
- QZ_COMPONENT_KERNEL_DRIVER
 - Data Compression API, 19
- QZ_COMPONENT_QATZIP_API
 - Data Compression API, 19
- QZ_COMPONENT_SOFTWARE_PROVIDER
 - Data Compression API, 19
- QZ_COMPONENT_USER_DRIVER
 - Data Compression API, 19
- QZ_COMPRESSED_SZ_OF_EMPTY_FILE
 - qatzip.h, 107
- QZ_CRC32
 - Data Compression API, 18
- QZ_CRC32_VALID
 - qatzip.h, 107
- QZ_CRC32_VALID_BIT
 - qatzip.h, 107
- QZ_CRC32_VALID_MASK
 - qatzip.h, 107
- QZ_CRC64_VALID
 - qatzip.h, 107
- QZ_CRC64_VALID_BIT
 - qatzip.h, 107
- QZ_CRC64_VALID_MASK
 - qatzip.h, 107
- QZ_DATA_ERROR
 - qatzip.h, 98
- QZ_DATA_FORMAT_DEFAULT
 - qatzip.h, 102
- QZ_DEFLATE
 - qatzip.h, 101
- QZ_DEFLATE_4B
 - Data Compression API, 17
- QZ_DEFLATE_COMP_LVL_MAXIMUM
 - qatzip.h, 104
- QZ_DEFLATE_COMP_LVL_MAXIMUM_Gen3
 - qatzip.h, 104
- QZ_DEFLATE_COMP_LVL_MINIMUM
 - qatzip.h, 104
- QZ_DEFLATE_GZIP
 - Data Compression API, 17
- QZ_DEFLATE_GZIP_EXT
 - Data Compression API, 17
- QZ_DEFLATE_RAW
 - Data Compression API, 17
- QZ_DIR_BOTH
 - Data Compression API, 17
- QZ_DIR_COMPRESS
 - Data Compression API, 17
- QZ_DIR_DECOMPRESS
 - Data Compression API, 17

- QZ_DIRECTION_DEFAULT
 - qatzip.h, [102](#)
- QZ_DISABLE_SOFTWARE_BACKUP
 - qatzip.h, [105](#)
- QZ_DISABLE_SOFTWARE_ONLY_EXECUTION
 - qatzip.h, [105](#)
- QZ_DUPLICATE
 - qatzip.h, [98](#)
- QZ_DYNAMIC_HDR
 - Data Compression API, [16](#)
- QZ_ENABLE_SOFTWARE_BACKUP
 - qatzip.h, [105](#)
- QZ_ENABLE_SOFTWARE_ONLY_EXECUTION
 - qatzip.h, [105](#)
- QZ_FAIL
 - qatzip.h, [98](#)
- QZ_FMT_NUM
 - Data Compression API, [17](#)
- QZ_FORCE_SW
 - qatzip.h, [98](#)
- QZ_HUFF_HDR_DEFAULT
 - qatzip.h, [102](#)
- QZ_HW_BUFF_MAX_SZ
 - qatzip.h, [103](#)
- QZ_HW_BUFF_MAX_SZ_Gen3
 - qatzip.h, [103](#)
- QZ_HW_BUFF_MIN_SZ
 - qatzip.h, [103](#)
- QZ_HW_BUFF_SZ
 - qatzip.h, [102](#)
- QZ_HW_BUFF_SZ_Gen3
 - qatzip.h, [102](#)
- QZ_HW_TIMEOUT
 - qatzip.h, [106](#)
- QZ_INPUT_CRC_VALID
 - qatzip.h, [107](#)
- QZ_INPUT_CRC_VALID_BIT
 - Data Compression API, [11](#)
- QZ_INPUT_CRC_VALID_MASK
 - qatzip.h, [106](#)
- QZ_INTEG
 - qatzip.h, [99](#)
- QZ_LOW_DEST_MEM
 - qatzip.h, [99](#)
- QZ_LOW_MEM
 - qatzip.h, [99](#)
- QZ_LZ4
 - qatzip.h, [101](#)
- QZ_LZ4_BLOCK
 - qatzip.h, [101](#)
- QZ_LZ4s
 - qatzip.h, [101](#)
- QZ_LZS_COMP_LVL_MAXIMUM
 - qatzip.h, [104](#)
- QZ_LZS_COMP_LVL_MINIMUM
 - qatzip.h, [104](#)
- QZ_MAX_ALGORITHMS
 - qatzip.h, [101](#)
- QZ_MAX_FORK_DEFAULT
 - qatzip.h, [102](#)
- QZ_MAX_STRING_LENGTH
 - Data Compression API, [10](#)
- QZ_METADATA_OVERFLOW
 - qatzip.h, [100](#)
- QZ_NO_HW
 - qatzip.h, [99](#)
- QZ_NO_INST_ATTACH
 - qatzip.h, [99](#)
- QZ_NO_MDRV
 - qatzip.h, [99](#)
- QZ_NO_SW_AVAIL
 - qatzip.h, [100](#)
- QZ_NONE
 - qatzip.h, [99](#)
- QZ_NOSW_LOW_MEM
 - qatzip.h, [100](#)
- QZ_NOSW_NO_HW
 - qatzip.h, [100](#)
- QZ_NOSW_NO_INST_ATTACH
 - qatzip.h, [100](#)
- QZ_NOSW_NO_MDRV
 - qatzip.h, [100](#)
- QZ_NOSW_UNSUPPORTED_FMT
 - qatzip.h, [100](#)
- QZ_NOT_SUPPORTED
 - qatzip.h, [101](#)
- QZ_OK
 - Data Compression API, [9](#)
- QZ_OUT_OF_RANGE
 - qatzip.h, [101](#)
- QZ_OUTPUT_CRC_VALID
 - qatzip.h, [107](#)
- QZ_OUTPUT_CRC_VALID_BIT
 - qatzip.h, [106](#)
- QZ_OUTPUT_CRC_VALID_MASK
 - qatzip.h, [107](#)
- QZ_PARAMS
 - qatzip.h, [98](#)
- QZ_PERIODICAL_POLLING
 - Data Compression API, [18](#)
- QZ_POLL_SLEEP_DEFAULT
 - qatzip.h, [102](#)
- QZ_POST_PROCESS_ERROR
 - qatzip.h, [100](#)
- QZ_POST_PROCESS_FAIL
 - qatzip.h, [106](#)
- QZ_POST_PROCESS_FAIL_BIT
 - qatzip.h, [106](#)
- QZ_POST_PROCESS_FAIL_MASK
 - qatzip.h, [106](#)
- QZ_REQ_THRESHOLD_DEFAULT
 - qatzip.h, [104](#)
- QZ_REQ_THRESHOLD_MAXIMUM
 - qatzip.h, [103](#)
- QZ_REQ_THRESHOLD_MINIMUM
 - qatzip.h, [103](#)

- QZ_SKID_PAD_SZ
 - Data Compression API, [11](#)
- QZ_STATIC_HDR
 - Data Compression API, [16](#)
- QZ_STRM_BUFF_MAX_SZ
 - qatzip.h, [103](#)
- QZ_STRM_BUFF_MIN_SZ
 - qatzip.h, [103](#)
- QZ_STRM_BUFF_SZ_DEFAULT
 - qatzip.h, [103](#)
- QZ_SW_BACKUP_BIT_POSITION
 - Data Compression API, [10](#)
- QZ_SW_BACKUP_DEFAULT
 - qatzip.h, [102](#)
- QZ_SW_EXECUTION
 - qatzip.h, [106](#)
- QZ_SW_EXECUTION_BIT
 - Data Compression API, [10](#)
- QZ_SW_EXECUTION_MASK
 - qatzip.h, [105](#)
- QZ_SW_FORCESW_BIT_POSITION
 - qatzip.h, [105](#)
- QZ_TIMEOUT
 - qatzip.h, [98](#)
- QZ_TIMEOUT_BIT
 - qatzip.h, [106](#)
- QZ_TIMEOUT_MASK
 - qatzip.h, [106](#)
- QZ_UNSUPPORTED_FMT
 - qatzip.h, [99](#)
- QZ_WAIT_CNT_THRESHOLD_DEFAULT
 - qatzip.h, [104](#)
- QZ_XXHASH
 - Data Compression API, [18](#)
- QZ_ZSTD
 - qatzip.h, [101](#)
- QZ_ZSTD_BLOCKSIZE_DEFAULT
 - qatzip.h, [105](#)
- QZ_ZSTD_COMP_LVL_MAXIMUM
 - qatzip.h, [104](#)
- QZ_ZSTD_COMP_LVL_MINIMUM
 - qatzip.h, [104](#)
- QZ_ZSTD_HIST_BUFF_DEFAULT
 - qatzip.h, [105](#)
- QZ_ZSTD_HW_BUFF_SZ
 - qatzip.h, [105](#)
- qzAllocateMetadata
 - Data Compression API, [48](#)
- qzAsyncCallbackFn
 - Data Compression API, [14](#)
- qzCallback
 - QzSessionParamsLZ4S_T, [86](#)
- qzCallback_external
 - QzSessionParamsLZ4S_T, [86](#)
- qzClose
 - Data Compression API, [40](#)
- qzCompress
 - Data Compression API, [24](#)
- qzCompress2
 - Data Compression API, [41](#)
- qzCompressCrc
 - Data Compression API, [25](#)
- qzCompressCrc64
 - qatzip.h, [109](#)
- qzCompressCrc64Ext
 - qatzip.h, [109](#)
- qzCompressCrcExt
 - qatzip.h, [109](#)
- qzCompressExt
 - qatzip.h, [108](#)
- qzCompressStream
 - Data Compression API, [53](#)
- qzCompressWithDictionary
 - Data Compression API, [29](#)
- qzCompressWithMetadataExt
 - Data Compression API, [27](#)
- QzCrc32Config_T, [73](#)
 - initial_value, [73](#)
 - polynomial, [73](#)
 - reflect_in, [74](#)
 - reflect_out, [74](#)
 - xor_out, [74](#)
- QzCrc64Config_T, [74](#)
 - initial_value, [75](#)
 - polynomial, [75](#)
 - reflect_in, [75](#)
 - reflect_out, [75](#)
 - xor_out, [75](#)
- QzCrcResult_T, [75](#)
 - crc_32, [76](#)
 - crc_64, [76](#)
 - in_crc, [76](#)
 - out_crc, [76](#)
 - status, [76](#)
 - valid_flags, [76](#)
- QzCrcType_T
 - Data Compression API, [18](#)
- QzDataFormat_T
 - Data Compression API, [17](#)
- qzDecompress
 - Data Compression API, [30](#)
- qzDecompress2
 - Data Compression API, [37](#)
- qzDecompressCrc
 - Data Compression API, [32](#)
- qzDecompressCrc64
 - qatzip.h, [110](#)
- qzDecompressCrc64Ext
 - qatzip.h, [110](#)
- qzDecompressCrcExt
 - qatzip.h, [109](#)
- qzDecompressExt
 - qatzip.h, [109](#)
- qzDecompressStream
 - Data Compression API, [54](#)
- qzDecompressWithDictionary

- Data Compression API, [34](#)
- qzDecompressWithMetadataExt
 - Data Compression API, [35](#)
- QzDictionaryParams_T, [77](#)
 - compressed, [77](#)
 - crc64Addr, [77](#)
 - dictId, [77](#)
 - dictionary, [77](#)
 - dictionary_len, [77](#)
- QzDirection_T
 - Data Compression API, [17](#)
- qzEndStream
 - Data Compression API, [56](#)
- qzFree
 - Data Compression API, [49](#)
- qzFreeMetadata
 - Data Compression API, [51](#)
- qzGetDefaults
 - Data Compression API, [46](#)
- qzGetDefaultsDeflate
 - qatzip.h, [111](#)
- qzGetDefaultsDeflateExt
 - qatzip.h, [112](#)
- qzGetDefaultsLZ4
 - qatzip.h, [111](#)
- qzGetDefaultsLZ4Block
 - qatzip.h, [111](#)
- qzGetDefaultsLZ4S
 - qatzip.h, [111](#)
- qzGetDefaultsZstd
 - qatzip.h, [111](#)
- qzGetDeflateEndOfStream
 - qatzip.h, [110](#)
- qzGetSessionCrc32Config
 - Data Compression API, [61](#)
- qzGetSessionCrc64Config
 - Data Compression API, [60](#)
- qzGetSoftwareComponentCount
 - Data Compression API, [59](#)
- qzGetSoftwareComponentVersionList
 - Data Compression API, [57](#)
- qzGetStatus
 - Data Compression API, [42](#)
- QzHuffmanHdr_T
 - Data Compression API, [15](#)
- qzInit
 - Data Compression API, [20](#)
- QzLogLevel_T
 - Data Compression API, [19](#)
- qzLZ4SCallbackFn
 - Data Compression API, [12](#)
- qzMalloc
 - Data Compression API, [47](#)
- qzMaxCompressedLength
 - qatzip.h, [110](#)
- qzMemFindAddr
 - Data Compression API, [52](#)
- QzMetadataBlob_T

- Data Compression API, [15](#)
- qzMetadataBlockGetCrc32
 - Data Compression API, [70](#)
- qzMetadataBlockGetCrc64
 - Data Compression API, [69](#)
- qzMetadataBlockRead
 - Data Compression API, [65](#)
- qzMetadataBlockWrite
 - Data Compression API, [67](#)
- QzPollingMode_T
 - Data Compression API, [18](#)
- QzResult_T, [77](#)
 - cb_tag, [78](#)
 - crc, [79](#)
 - dest_len, [78](#)
 - ext_rc, [78](#)
 - extension_result, [79](#)
 - src_len, [78](#)
 - status, [78](#)
- QzSession_T, [79](#)
 - hw_session_stat, [79](#)
 - internal, [80](#)
 - thd_sess_stat, [79](#)
 - total_in, [80](#)
 - total_out, [80](#)
- QzSessionParams_T, [80](#)
 - comp_algorithm, [81](#)
 - comp_lvl, [81](#)
 - data_fmt, [81](#)
 - direction, [81](#)
 - huffman_hdr, [81](#)
 - hw_buff_sz, [82](#)
 - input_sz_thrshold, [82](#)
 - max_forks, [81](#)
 - req_cnt_thrshold, [82](#)
 - strm_buff_sz, [82](#)
 - sw_backup, [81](#)
 - wait_cnt_thrshold, [82](#)
- QzSessionParamsCommon_T, [82](#)
 - comp_algorithm, [83](#)
 - comp_lvl, [83](#)
 - direction, [83](#)
 - hw_buff_sz, [83](#)
 - input_sz_thrshold, [84](#)
 - is_sensitive_mode, [84](#)
 - max_forks, [83](#)
 - polling_mode, [84](#)
 - req_cnt_thrshold, [84](#)
 - strm_buff_sz, [84](#)
 - sw_backup, [83](#)
 - wait_cnt_thrshold, [84](#)
- QzSessionParamsDeflate_T, [84](#)
 - common_params, [85](#)
 - data_fmt, [85](#)
 - huffman_hdr, [85](#)
- QzSessionParamsDeflateExt_T, [85](#)
 - deflate_params, [85](#)
 - stop_decompression_stream_end, [85](#)

- zlib_format, 85
- QzSessionParamsLZ4_T, 86
 - common_params, 86
- QzSessionParamsLZ4S_T, 86
 - common_params, 86
 - lz4s_mini_match, 87
 - qzCallback, 86
 - qzCallback_external, 86
- QzSessionParamsZstd_T, 87
 - blockSize, 87
 - common_params, 87
 - historyBuffer, 87
 - reserved, 87
- qzSetDefaults
 - Data Compression API, 44
- qzSetDefaultsDeflate
 - qatzip.h, 110
- qzSetDefaultsDeflateExt
 - qatzip.h, 111
- qzSetDefaultsLZ4
 - qatzip.h, 110
- qzSetDefaultsLZ4Block
 - qatzip.h, 111
- qzSetDefaultsLZ4S
 - qatzip.h, 111
- qzSetDefaultsZstd
 - qatzip.h, 111
- qzSetLogLevel
 - Data Compression API, 19
- qzSetSessionCrc32Config
 - Data Compression API, 64
- qzSetSessionCrc64Config
 - Data Compression API, 63
- qzSetupSession
 - Data Compression API, 22
- qzSetupSessionDeflate
 - qatzip.h, 108
- qzSetupSessionDeflateExt
 - qatzip.h, 108
- qzSetupSessionLZ4
 - qatzip.h, 108
- qzSetupSessionLZ4Block
 - qatzip.h, 108
- qzSetupSessionLZ4S
 - qatzip.h, 108
- qzSetupSessionZstd
 - qatzip.h, 108
- QzSoftwareComponentType_T
 - Data Compression API, 18
- QzSoftwareVersionInfo_T, 88
 - build_number, 88
 - component_name, 88
 - component_type, 88
 - major_version, 88
 - minor_version, 88
 - patch_version, 88
 - reserved, 88
- QzStatus_T, 89
 - algo_hw, 90
 - algo_sw, 90
 - hw_session_status, 90
 - memory_allocated, 90
 - qat_hw_count, 89
 - qat_instance_attach, 89
 - qat_mem_drvr, 89
 - qat_service_init, 89
 - using_huge_pages, 90
- QzStream_T, 90
 - crc_32, 92
 - crc_type, 92
 - in, 91
 - in_sz, 91
 - opaque, 92
 - out, 91
 - out_sz, 91
 - pending_in, 91
 - pending_out, 92
 - reserved, 92
- qzTeardownSession
 - Data Compression API, 38
- reflect_in
 - QzCrc32Config_T, 74
 - QzCrc64Config_T, 75
- reflect_out
 - QzCrc32Config_T, 74
 - QzCrc64Config_T, 75
- req_cnt_thrsld
 - QzSessionParams_T, 82
 - QzSessionParamsCommon_T, 84
- reserved
 - QzSessionParamsZstd_T, 87
 - QzSoftwareVersionInfo_T, 88
 - QzStream_T, 92
- src_len
 - QzResult_T, 78
- status
 - QzCrcResult_T, 76
 - QzResult_T, 78
- stop_decompression_stream_end
 - QzSessionParamsDeflateExt_T, 85
- strm_buff_sz
 - QzSessionParams_T, 82
 - QzSessionParamsCommon_T, 84
- sw_backup
 - QzSessionParams_T, 81
 - QzSessionParamsCommon_T, 83
- thd_sess_stat
 - QzSession_T, 79
- total_in
 - QzSession_T, 80
- total_out
 - QzSession_T, 80
- using_huge_pages

- [QzStatus_T](#), [90](#)
- [valid_flags](#)
 - [QzCrcResult_T](#), [76](#)
- [wait_cnt_threshold](#)
 - [QzSessionParams_T](#), [82](#)
 - [QzSessionParamsCommon_T](#), [84](#)
- [xor_out](#)
 - [QzCrc32Config_T](#), [74](#)
 - [QzCrc64Config_T](#), [75](#)
- [zlib_format](#)
 - [QzSessionParamsDeflateExt_T](#), [85](#)