Naples Offload API Reference Guide v. 0.1

Generated from SONIC driver version 1.0

Contents

Chapter 1

Data Structure Index

1.1 Data Structures

Here are the data structures with brief descriptions:

Chapter 2

File Index

2.1 File List

Here is a list of all files with brief descriptions:

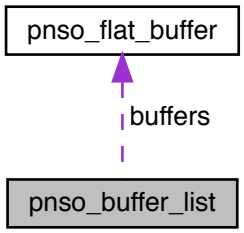
Chapter 3

Data Structure Documentation

3.1 pnso_buffer_list Struct Reference

Describes a scatter/gather buffer list.

Collaboration diagram for pnso_buffer_list:



Data Fields

- uint32_t count
- struct pnso_flat_buffer buffers [0]

3.1.1 Detailed Description

Describes a scatter/gather buffer list.

struct pnso_buffer_list - describes a scatter/gather buffer list.

Parameters

count	specifies the number of buffers in the list.
buffers	specifies an unbounded array of flat buffers as defined by 'count'.

This structure is typically used to represent a collection of physical memory buffers that are not contiguous.

3.1.2 Field Documentation

3.1.2.1 buffers

```
struct pnso_flat_buffer pnso_buffer_list::buffers[0]
```

3.1.2.2 count

```
uint32_t pnso_buffer_list::count
```

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

- pnso_api.h

3.2 pnso_checksum_desc Struct Reference

Represents the descriptor for checksum operation.

Data Fields

- uint16_t algo_type
- uint16_t flags

3.2.1 Detailed Description

Represents the descriptor for checksum operation.

struct pnso_checksum_desc - represents the descriptor for checksum operation.

Parameters

algo_type	specifies one of the enumerated values of the checksum algorithm (i.e. pnso_chksum_type).
flags	specifies the following applicable descriptor flag(s) to checksum descriptor. PNSO_CHKSUM_DFLAG_PER_BLOCK - indicates to produce one checksum per block. When this flag is not specified, checksum for the entire buffer will be produced.

3.2.2 Field Documentation

3.2.2.1 algo_type

```
uint16_t pnso_checksum_desc::algo_type
```

3.2.2.2 flags

```
uint16_t pnso_checksum_desc::flags
```

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

- pnso_api.h

3.3 pnso_chksum_tag Struct Reference

Represents the checksum tag.

Data Fields

- uint8_t chksum [PNSO_CHKSUM_TAG_LEN]

3.3.1 Detailed Description

Represents the checksum tag.

struct pnso_chksum_tag - represents the checksum tag.

Parameters

chksum	specifies 4 or 8-byte checksum.

3.3.2 Field Documentation

3.3.2.1 chksum

```
uint8_t pnso_chksum_tag::chksum[PNSO_CHKSUM_TAG_LEN]
```

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

- pnso_api.h

3.4 pnso_compression_desc Struct Reference

Represents the descriptor for compression service.

Data Fields

- uint16_t algo_type
- uint16_t flags
- uint16_t threshold_len
- uint16_t hdr_fmt_idx
- uint32_t hdr_algo

3.4.1 Detailed Description

Represents the descriptor for compression service.

struct pnso_compression_desc - represents the descriptor for compression service.

Parameters

algo_type	specifies one of the enumerated values of the compressor algorithm (i.e. pnso_compressor_type).
flags	specifies the following applicable descriptor flags to compression descriptor. PNSO_CP_DFLAG_ZERO_PAD - indicates to zero fill the compressed output buffer aligning to block size.

PNSO_CP_DFLAG_INSERT_HEADER - indicates to insert compression header defined by the format supplied in 'struct pnso_init_params'.

PNSO_CP_DFLAG_BYPASS_ONFAIL - indicates to use the source buffer as input buffer to hash and/or checksum, services, when compression operation fails. This flag is effective only when compression, hash and/or checksum operation is requested.

Parameters

threshold_len	specifies the expected compressed buffer length in bytes. This is to instruct the compression operation, upon its completion, to compress the buffer to a length that must be less than or equal to 'threshold_len'.
hdr_fmt_idx	specifies the index for the header format in the header format array.
hdr_algo	specifies the value for header field PNSO_HDR_FIELD_TYPE_ALGO. This is the same value that is registered in pnso_add_compression_algo_mapping.

3.4.2 Field Documentation

3.4.2.1 algo_type

```
uint16_t pnso_compression_desc::algo_type
```

3.4.2.2 flags

```
uint16_t pnso_compression_desc::flags
```

3.4.2.3 hdr_algo

```
uint32_t pnso_compression_desc::hdr_algo
```

3.4.2.4 hdr_fmt_idx

```
uint16_t pnso_compression_desc::hdr_fmt_idx
```

3.4.2.5 threshold_len

```
uint16_t pnso_compression_desc::threshold_len
```

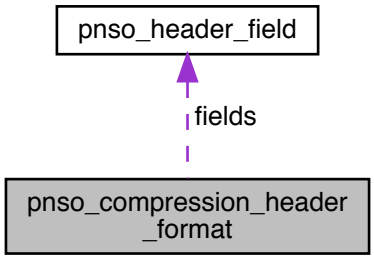
The documentation for this struct was generated from the following file:

- pnso_api.h

3.5 pnso_compression_header_format Struct Reference

Represents the format of the compression header.

Collaboration diagram for pnso_compression_header_format:



Data Fields

- uint32_t num_fields
- struct pnso_header_field fields [PNSO_MAX_HEADER_FIELDS]

3.5.1 Detailed Description

Represents the format of the compression header.

struct pnso_compression_header_format - represents the format of the compression header.

Parameters

num_fields	specifies the number of fields in the bounded array.
fields	specifies an array of fields.

3.5.2 Field Documentation

3.5.2.1 fields

```
struct pnso_header_field pnso_compression_header_format::fields[PNSO_MAX_HEADER_FIELDS]
```

3.5.2.2 num_fields

```
uint32_t pnso_compression_header_format::num_fields
```

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

- pnso_api.h

3.6 pnso_crypto_desc Struct Reference

Represents the descriptor for encryption or decryption operation.

Data Fields

- uint16_t algo_type
- uint16_t rsvd
- uint32_t key_desc_idx
- uint64_t iv_addr

3.6.1 Detailed Description

Represents the descriptor for encryption or decryption operation.

struct pnso_crypto_desc - represents the descriptor for encryption or decryption operation.

Parameters

algo_type	specifies one of the enumerated values of the crypto type (i.e. pnso_crypto_type).
rsvd	specifies a 'reserved' field meant to be used by Pensando.
key_desc_idx	specifies the key index in the descriptor table.
iv_addr	specifies the physical address of the initialization vector.

3.6.2 Field Documentation

3.6.2.1 algo_type

```
uint16_t pnso_crypto_desc::algo_type
```

3.6.2.2 iv_addr

```
uint64_t pnso_crypto_desc::iv_addr
```

3.6.2.3 key_desc_idx

```
uint32_t pnso_crypto_desc::key_desc_idx
```

3.6.2.4 rsvd

```
uint16_t pnso_crypto_desc::rsvd
```

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

- pnso_api.h

3.7 pnso_decompaction_desc Struct Reference

Represents the descriptor for decompaction operation.

Data Fields

- uint64_t vvbn:48
- uint64_t rsvd_1:16
- uint16_t hdr_fmt_idx
- uint16_t rsvd_2

3.7.1 Detailed Description

Represents the descriptor for decompaction operation.

struct pnso_decompaction_desc - represents the descriptor for decompaction operation.

Parameters

vvbn	specifies the block number within the Netapp's packed block header, with which the offset and length of data can be retrieved.
hdr_fmt_idx	compression header format used for decoding any compressed block in the compacted region.

3.7.2 Field Documentation

3.7.2.1 hdr_fmt_idx

```
uint16_t pnso_decompaction_desc::hdr_fmt_idx
```

3.7.2.2 rsvd_1

```
uint64_t pnso_decompaction_desc::rsvd_1
```

3.7.2.3 rsvd_2

```
uint16_t pnso_decompaction_desc::rsvd_2
```

3.7.2.4 vvbn

```
uint64_t pnso_decompaction_desc::vvbn
```

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

- pnso_api.h

3.8 pnso_decompression_desc Struct Reference

Represents the descriptor for decompression operation.

Data Fields

- uint16_t algo_type
- uint16_t flags
- uint16_t hdr_fmt_idx
- uint16_t rsvd

3.8.1 Detailed Description

Represents the descriptor for decompression operation.

struct pnso_decompression_desc - represents the descriptor for decompression operation.

Parameters

algo_type	specifies one of the enumerated values of the compressor algorithm (i.e. pnso_compressor_type) for decompression.
flags	specifies the following applicable descriptor flags to decompression descriptor. PNSO_DC_DFLAG_HEADER_PRESENT - indicates the compression header is present.
hdr_fmt_idx	specifies the index for the header format in the header format array.
rsvd	specifies a 'reserved' field meant to be used by Pensando.

3.8.2 Field Documentation

3.8.2.1 algo_type

```
uint16_t pnso_decompression_desc::algo_type
```

3.8.2.2 flags

```
uint16_t pnso_decompression_desc::flags
```

3.8.2.3 hdr_fmt_idx

```
uint16_t pnso_decompression_desc::hdr_fmt_idx
```

3.8.2.4 rsvd

```
uint16_t pnso_decompression_desc::rsvd
```

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

- pnso_api.h

3.9 pnso_flat_buffer Struct Reference

Describes a buffer with 'address and length'.

Data Fields

- uint32_t len
- uint64_t buf

3.9.1 Detailed Description

Describes a buffer with 'address and length'.

struct pnso_flat_buffer - describes a buffer with 'address and length'.

Parameters

len	contains the length of the buffer in bytes.
buf	contains the physical address of a buffer.

3.9.2 Field Documentation

3.9.2.1 buf

```
uint64_t pnso_flat_buffer::buf
```

3.9.2.2 len

```
uint32_t pnso_flat_buffer::len
```

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

- pnso_api.h

3.10 pnso_hash_desc Struct Reference

Represents the descriptor for data deduplication hash operation.

Data Fields

- uint16_t algo_type
- uint16_t flags

3.10.1 Detailed Description

Represents the descriptor for data deduplication hash operation.

struct pnso_hash_desc - represents the descriptor for data deduplication hash operation.

Parameters

algo_type	specifies one of the enumerated values of the hash algorithm (i.e. pnso_hash_type) for data deduplication.
flags	specifies the following applicable descriptor flag(s) to hash descriptor. PNSO_HASH_DFLAG_PER_BLOCK - indicates to produce one hash per block. When this flag is not specified, hash for the entire buffer will be produced.

3.10.2 Field Documentation

3.10.2.1 algo_type

```
uint16_t pnso_hash_desc::algo_type
```

3.10.2.2 flags

```
uint16_t pnso_hash_desc::flags
```

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

- pnso_api.h

3.11 pnso_hash_tag Struct Reference

Represents the SHA hash tag.

Data Fields

- uint8_t hash [PNSO_HASH_TAG_LEN]

3.11.1 Detailed Description

Represents the SHA hash tag.

struct pnso_hash_tag - represents the SHA hash tag.

Parameters

hash	specifies a 64 or 32-byte hash.

3.11.2 Field Documentation

3.11.2.1 hash

```
uint8_t pnso_hash_tag::hash[PNSO_HASH_TAG_LEN]
```

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

- pnso_api.h

3.12 pnso_header_field Struct Reference

Defines the value for each field in the compression header.

Data Fields

- enum pnso_header_field_type type
- uint32_t offset
- uint32_t length
- uint32_t value

3.12.1 Detailed Description

Defines the value for each field in the compression header.

struct pnso_header_field - defines the value for each field in the compression header.

Parameters

type	specifies the source for the header fields. Refer to 'enum pnso_header_field_type' section for more details on the type.
offset	specifies the offset of the value from the beginning of the header.
length	specifies the length of the value.
value	specifies the value.

3.12.2 Field Documentation

3.12.2.1 length

```
uint32_t pnso_header_field::length
```

3.12.2.2 offset

```
uint32_t pnso_header_field::offset
```

3.12.2.3 type

```
enum pnso_header_field_type pnso_header_field::type
```

3.12.2.4 value

```
uint32_t pnso_header_field::value
```

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

- pnso_api.h

3.13 pnso_init_params Struct Reference

Represents the initialization parameters for Pensando accelerators.

Data Fields

- uint16_t per_core_qdepth
- uint32_t block_size

3.13.1 Detailed Description

Represents the initialization parameters for Pensando accelerators.

struct pnso_init_params - represents the initialization parameters for Pensando accelerators.

Parameters

per_core_qdepth	specifies the maximum number of parallel requests per core.
block_size	specifies the size of a block in bytes.

3.13.2 Field Documentation

3.13.2.1 block_size

```
uint32_t pnso_init_params::block_size
```

3.13.2.2 per_core_qdepth

`uint16_t pnso_init_params::per_core_qdepth`

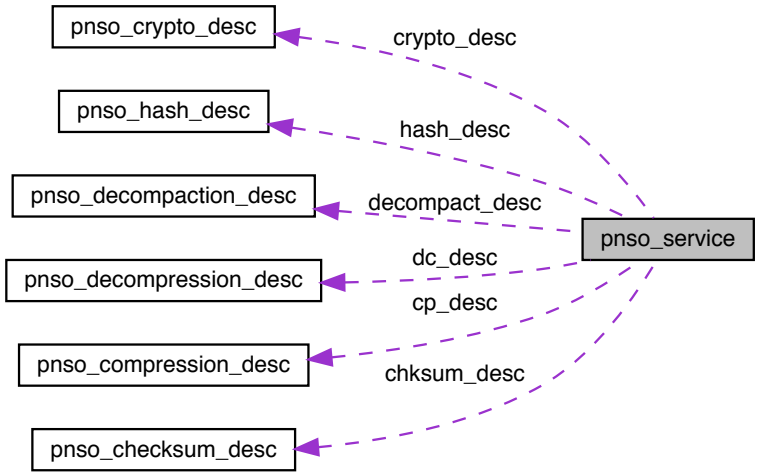
The documentation for this struct was generated from the following file:

- pnso_api.h

3.14 pnso_service Struct Reference

Describes various parameters of the service chosen for acceleration.

Collaboration diagram for pnso_service:



Data Fields

- uint16_t svc_type
- uint16_t rsvd
- union {
  struct pnso_crypto_desc crypto_desc
  struct pnso_compression_desc cp_desc
  struct pnso_decompression_desc dc_desc
  struct pnso_hash_desc hash_desc
  struct pnso_checksum_desc chksum_desc
  struct pnso_decompaction_desc decompact_desc
  } u

3.14.1 Detailed Description

Describes various parameters of the service chosen for acceleration.

struct pnso_service - describes various parameters of the service chosen for acceleration.

Parameters

svc_type	specifies one of the enumerated values for the accelerator service type.
crypto_desc	specifies the descriptor for encryption/decryption service.
cp_desc	specifies the descriptor for compression service.
dc_desc	specifies the descriptor for decompression service.
hash_desc	specifies the descriptor for deduplication service.
chksum_desc	specifies the descriptor for checksum service.
decompact_desc	specifies the descriptor for decompaction service.

3.14.2 Field Documentation

3.14.2.1 chksum_desc

```
struct pnso_checksum_desc pnso_service::chksum_desc
```

3.14.2.2 cp_desc

```
struct pnso_compression_desc pnso_service::cp_desc
```

3.14.2.3 crypto_desc

```
struct pnso_crypto_desc pnso_service::crypto_desc
```

3.14.2.4 dc_desc

```
struct pnso_decompression_desc pnso_service::dc_desc
```

3.14.2.5 decompact_desc

```
struct pnso_decompaction_desc pnso_service::decompact_desc
```

3.14.2.6 hash_desc

```
struct pnso_hash_desc pnso_service::hash_desc
```

3.14.2.7 rsvd

```
uint16_t pnso_service::rsvd
```

3.14.2.8 svc_type

```
uint16_t pnso_service::svc_type
```

3.14.2.9 u

```
union { ... } pnso_service::u
```

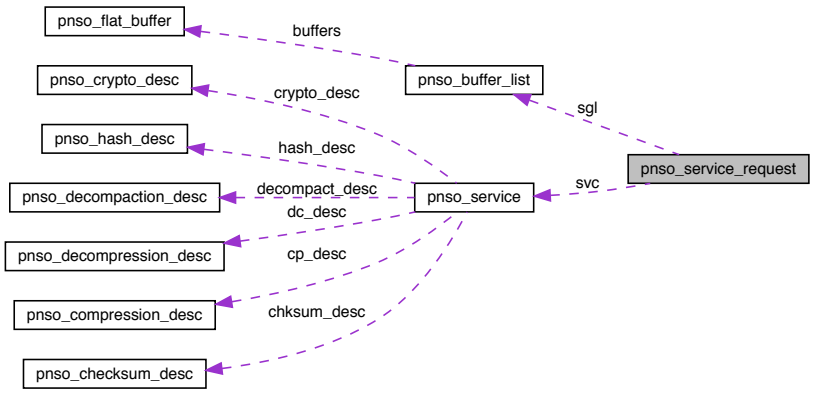
The documentation for this struct was generated from the following file:

- pnso_api.h

3.15 pnso_service_request Struct Reference

Represents an array of services that are to be handled in a request.

Collaboration diagram for pnso_service_request:



Data Fields

- struct pnso_buffer_list * sgl
- uint32_t num_services
- struct pnso_service svc [0]

3.15.1 Detailed Description

Represents an array of services that are to be handled in a request.

struct pnso_service_request - represents an array of services that are to be handled in a request.

Parameters

sgl	specifies input buffer list on which the request will operate on.
num_services	specifies the number of services in the input service request.
svc	specifies the information about each service within the service request.

3.15.2 Field Documentation

3.15.2.1 num_services

```
uint32_t pnso_service_request::num_services
```

3.15.2.2 sgl

```
struct pnso_buffer_list* pnso_service_request::sgl
```

3.15.2.3 svc

```
struct pnso_service pnso_service_request::svc[0]
```

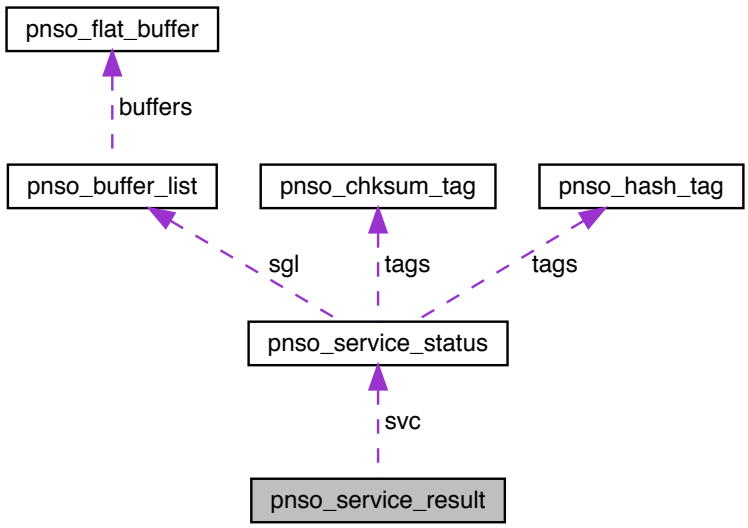
The documentation for this struct was generated from the following file:

- pnso_api.h

3.16 pnso_service_result Struct Reference

Represents the result of the request upon completion one or all services.

Collaboration diagram for pnso_service_result:



Data Fields

- pnso_error_t err
- uint32_t num_services
- struct pnso_service_status svc [0]

3.16.1 Detailed Description

Represents the result of the request upon completion one or all services.

struct pnso_service_result - represents the result of the request upon completion one or all services.

Parameters

err	specifies the overall error code of the request. When set to '0', the request processing can be considered successful. Otherwise, one of the services in the request is failed, and any output data should be discarded.
num_services	specifies the number of services in the request.
svc	specifies an array of service status structures to report the status of each service within a request upon its completion.

When 'err' is set to '0', the overall request processing can be considered successful. Otherwise, one of the services

in the request is failed, and any output data should be discarded.

3.16.2 Field Documentation

3.16.2.1 err

```
pnso_error_t pnso_service_result::err
```

3.16.2.2 num_services

```
uint32_t pnso_service_result::num_services
```

3.16.2.3 svc

```
struct pnso_service_status pnso_service_result::svc[0]
```

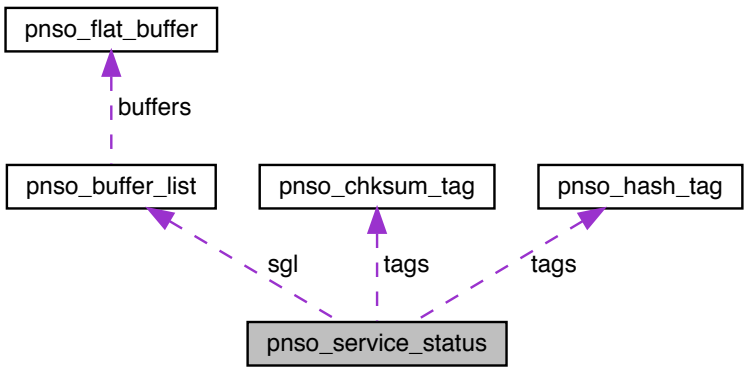
The documentation for this struct was generated from the following file:

- pnso_api.h

3.17 pnso_service_status Struct Reference

Represents the result of a specific service within a request.

Collaboration diagram for pnso_service_status:



Data Fields

- pnso_error_t err
- uint16_t svc_type
- uint16_t rsvd_1
- union {
  struct {
  uint16_t num_tags
  uint16_t rsvd_2
  struct pnso_hash_tag * tags
  } hash
  struct {
  uint16_t num_tags
  uint16_t rsvd_3
  struct pnso_chksum_tag * tags
  } chksum
  struct {
  uint32_t data_len
  struct pnso_buffer_list * sgl
  } dst
  } u

3.17.1 Detailed Description

Represents the result of a specific service within a request.

struct pnso_service_status - represents the result of a specific service within a request.

Parameters

err	specifies the error code of a service within the service request. When 'err' is set to '0', the processing of this service can be considered successful. Otherwise, this service in the request is failed, and any output data should be discarded.
svc_type	specifies one of the enumerated values for the accelerator service type.
rsvd_1	specifies a 'reserved' field meant to be used by Pensando.
hash	specifies a pointer to an allocated memory for number of hashes as specified in 'num_tags'. When 'num_tags' is 0, this parameter is NULL.
chksum	specifies a pointer to an allocated memory for number of checksums as specified in 'num_tags'. When 'num_tags' is 0, this parameter is NULL.
dst	specifies a sgl that to be used as output buffer for this service. 'data_len' specifies the length of the data within the the sgl.

Note: Hash or checksum tags will be packed one after another.

3.17.2 Field Documentation

3.17.2.1 chksum

```
struct { ... } pnso_service_status::chksum
```

3.17.2.2 data_len

```
uint32_t pnso_service_status::data_len
```

3.17.2.3 dst

```
struct { ... } pnso_service_status::dst
```

3.17.2.4 err

```
pnso_error_t pnso_service_status::err
```

3.17.2.5 hash

```
struct { ... } pnso_service_status::hash
```

3.17.2.6 num_tags

```
uint16_t pnso_service_status::num_tags
```

3.17.2.7 rsvd_1

```
uint16_t pnso_service_status::rsvd_1
```

3.17.2.8 rsvd_2

```
uint16_t pnso_service_status::rsvd_2
```

3.17.2.9 rsvd_3

```
uint16_t pnso_service_status::rsvd_3
```

3.17.2.10 sgl

```
struct pnso_buffer_list* pnso_service_status::sgl
```

3.17.2.11 svc_type

```
uint16_t pnso_service_status::svc_type
```

3.17.2.12 tags [1/2]

```
struct pnso_hash_tag* pnso_service_status::tags
```

3.17.2.13 tags [2/2]

```
struct pnso_chksum_tag* pnso_service_status::tags
```

3.17.2.14 u

```
union { ... } pnso_service_status::u
```

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

- pnso_api.h

Chapter 4

File Documentation

4.1 pnso_api.h File Reference

Data Structures

- struct pnso_flat_buffer

  Describes a buffer with 'address and length'.
- struct pnso_buffer_list

  Describes a scatter/gather buffer list.
- struct pnso_header_field

  Defines the value for each field in the compression header.
- struct pnso_compression_header_format

  Represents the format of the compression header.
- struct pnso_init_params

  Represents the initialization parameters for Pensando accelerators.
- struct pnso_crypto_desc

  Represents the descriptor for encryption or decryption operation.
- struct pnso_compression_desc

  Represents the descriptor for compression service.
- struct pnso_decompression_desc

  Represents the descriptor for decompression operation.
- struct pnso_hash_desc

  Represents the descriptor for data deduplication hash operation.
- struct pnso_checksum_desc

  Represents the descriptor for checksum operation.
- struct pnso_decompaction_desc

  Represents the descriptor for decompaction operation.
- struct pnso_hash_tag

  Represents the SHA hash tag.
- struct pnso_chksum_tag

  Represents the checksum tag.
- struct pnso_service_status

  Represents the result of a specific service within a request.
- struct pnso_service_result

  Represents the result of the request upon completion one or all services.
- struct pnso_service

  Describes various parameters of the service chosen for acceleration.
- struct pnso_service_request

  Represents an array of services that are to be handled in a request.

Macros

- #define PNSO_OK 0
- #define PNSO_ERR_CPDC_AXI_TIMEOUT 20001
- #define PNSO_ERR_CPDC_AXI_DATA_ERROR 20002
- #define PNSO_ERR_CPDC_AXI_ADDR_ERROR 20003
- #define PNSO_ERR_CPDC_COMPRESSION_FAILED 20004
- #define PNSO_ERR_CPDC_DATA_TOO_LONG 20005
- #define PNSO_ERR_CPDC_CHECKSUM_FAILED 20006
- #define PNSO_ERR_CPDC_SGL_DESC_ERROR 20007
- #define PNSO_ERR_CPDC_HDR_IDX_INVALID 20008
- #define PNSO_ERR_CPDC_ALGO_INVALID 20009
- #define PNSO_ERR_CRYPTO_KEY_INDEX_OUT_OF_RANG 30001
- #define PNSO_ERR_CRYPTO_WRONG_KEY_TYPE 30002
- #define PNSO_ERR_CRYPTO_AXI_ERROR 30003
- #define PNSO_ERR_CRYPTO_AXI_STATUS_ERROR 30004
- #define PNSO_ERR_CRYPTO_AOL_DESC_ERROR 30005
- #define PNSO_ERR_CRYPTO_KEY_NOT_REGISTERED 30006
- #define PNSO_ERR_CRYPTO_LEN_NOT_MULTI_SECTORS 30007
- #define PNSO_ERR_CRYPTO_GENERAL_ERROR 30008
- #define PNSO_ERR_SHA_FAILED 40001
- #define PNSO_MAX_HEADER_FIELDS 8
- #define PNSO_CP_DFLAG_ZERO_PAD (1 << 0)
- #define PNSO_CP_DFLAG_INSERT_HEADER (1 << 1)
- #define PNSO_CP_DFLAG_BYPASS_ONFAIL (1 << 2)
- #define PNSO_DC_DFLAG_HEADER_PRESENT (1 << 0)
- #define PNSO_HASH_DFLAG_PER_BLOCK (1 << 0)
- #define PNSO_CHKSUM_DFLAG_PER_BLOCK (1 << 0)
- #define PNSO_HASH_TAG_LEN 64
- #define PNSO_CHKSUM_TAG_LEN 8

Typedefs

- typedef int32_t pnso_error_t
- typedef void(* completion_cb_t) (void *cb_ctx, struct pnso_service_result *svc_res)
- typedef pnso_error_t(* pnso_poll_fn_t) (void *pnso_poll_ctx)

Enumerations

- enum pnso_service_type {
  PNSO_SVC_TYPE_NONE = 0,
  PNSO_SVC_TYPE_ENCRYPT = 1,
  PNSO_SVC_TYPE_DECRYPT = 2,
  PNSO_SVC_TYPE_COMPRESS = 3,
  PNSO_SVC_TYPE_DECOMPRESS = 4,
  PNSO_SVC_TYPE_HASH = 5,
  PNSO_SVC_TYPE_CHKSUM = 6,
  PNSO_SVC_TYPE_DECOMPACT = 7,
  PNSO_SVC_TYPE_MAX }
- enum pnso_crypto_type {
  PNSO_CRYPTO_TYPE_NONE = 0,
  PNSO_CRYPTO_TYPE_XTS = 1,
  PNSO_CRYPTO_TYPE_GCM = 2,
  PNSO_CRYPTO_TYPE_MAX }

- enum pnso_compression_type {
  PNSO_COMPRESSION_TYPE_NONE = 0,
  PNSO_COMPRESSION_TYPE_LZRW1A = 1,
  PNSO_COMPRESSION_TYPE_MAX }
- enum pnso_hash_type {
  PNSO_HASH_TYPE_NONE = 0,
  PNSO_HASH_TYPE_SHA2_512 = 1,
  PNSO_HASH_TYPE_SHA2_256 = 2,
  PNSO_HASH_TYPE_MAX }
- enum pnso_chksum_type {
  PNSO_CHKSUM_TYPE_NONE = 0,
  PNSO_CHKSUM_TYPE_MCRC64 = 1,
  PNSO_CHKSUM_TYPE_CRC32C = 2,
  PNSO_CHKSUM_TYPE_ADLER32 = 3,
  PNSO_CHKSUM_TYPE_MADLER32 = 4,
  PNSO_CHKSUM_TYPE_MAX }
- enum pnso_header_field_type {
  PNSO_HDR_FIELD_TYPE_NONE = 0,
  PNSO_HDR_FIELD_TYPE_STATIC = 1,
  PNSO_HDR_FIELD_TYPE_INDATA_CHKSUM = 2,
  PNSO_HDR_FIELD_TYPE_OUTDATA_LENGTH = 3,
  PNSO_HDR_FIELD_TYPE_ALGO = 4,
  PNSO_HDR_FIELD_TYPE_MAX }

Functions

- pnso_error_t pnso_init (struct pnso_init_params *init_params)
- pnso_error_t pnso_submit_request (struct pnso_service_request *svc_req, struct pnso_service_result *svc_res, completion_cb_t cb, void *cb_ctx, pnso_poll_fn_t *pnso_poll_fn, void **pnso_poll_ctx)
- pnso_error_t pnso_add_to_batch (struct pnso_service_request *svc_req, struct pnso_service_result *svc_res)
- pnso_error_t pnso_flush_batch (completion_cb_t cb, void *cb_ctx, pnso_poll_fn_t *pnso_poll_fn, void **pnso_poll_ctx)
- pnso_error_t pnso_set_key_desc_idx (const void *key1, const void *key2, uint32_t key_size, uint32_t key_idx)
- pnso_error_t pnso_register_compression_header_format (struct pnso_compression_header_format *cp_hdr_fmt, uint16_t hdr_fmt_idx)
- pnso_error_t pnso_add_compression_algo_mapping (enum pnso_compression_type pnso_algo, uint32_t header_algo)

4.1.1 Macro Definition Documentation

4.1.1.1 PNSO_CHKSUM_DFLAG_PER_BLOCK

```
#define PNSO_CHKSUM_DFLAG_PER_BLOCK (1 << 0)
```

4.1.1.2 PNSO_CHKSUM_TAG_LEN

```
#define PNSO_CHKSUM_TAG_LEN 8
```

4.1.1.3 PNSO_CP_DFLAG_BYPASS_ONFAIL

```
#define PNSO_CP_DFLAG_BYPASS_ONFAIL (1 << 2)
```

4.1.1.4 PNSO_CP_DFLAG_INSERT_HEADER

```
#define PNSO_CP_DFLAG_INSERT_HEADER (1 << 1)
```

4.1.1.5 PNSO_CP_DFLAG_ZERO_PAD

```
#define PNSO_CP_DFLAG_ZERO_PAD (1 << 0)
```

4.1.1.6 PNSO_DC_DFLAG_HEADER_PRESENT

```
#define PNSO_DC_DFLAG_HEADER_PRESENT (1 << 0)
```

4.1.1.7 PNSO_ERR_CPDC_ALGO_INVALID

```
#define PNSO_ERR_CPDC_ALGO_INVALID 20009
```

4.1.1.8 PNSO_ERR_CPDC_AXI_ADDR_ERROR

```
#define PNSO_ERR_CPDC_AXI_ADDR_ERROR 20003
```

4.1.1.9 PNSO_ERR_CPDC_AXI_DATA_ERROR

```
#define PNSO_ERR_CPDC_AXI_DATA_ERROR 20002
```

4.1.1.10 PNSO_ERR_CPDC_AXI_TIMEOUT

```
#define PNSO_ERR_CPDC_AXI_TIMEOUT 20001
```

4.1.1.11 PNSO_ERR_CPDC_CHECKSUM_FAILED

```
#define PNSO_ERR_CPDC_CHECKSUM_FAILED 20006
```

4.1.1.12 PNSO_ERR_CPDC_COMPRESSION_FAILED

```
#define PNSO_ERR_CPDC_COMPRESSION_FAILED 20004
```

4.1.1.13 PNSO_ERR_CPDC_DATA_TOO_LONG

```
#define PNSO_ERR_CPDC_DATA_TOO_LONG 20005
```

4.1.1.14 PNSO_ERR_CPDC_HDR_IDX_INVALID

```
#define PNSO_ERR_CPDC_HDR_IDX_INVALID 20008
```

4.1.1.15 PNSO_ERR_CPDC_SGL_DESC_ERROR

```
#define PNSO_ERR_CPDC_SGL_DESC_ERROR 20007
```

4.1.1.16 PNSO_ERR_CRYPTO_AOL_DESC_ERROR

```
#define PNSO_ERR_CRYPTO_AOL_DESC_ERROR 30005
```

4.1.1.17 PNSO_ERR_CRYPTO_AXI_ERROR

```
#define PNSO_ERR_CRYPTO_AXI_ERROR 30003
```

4.1.1.18 PNSO_ERR_CRYPTO_AXI_STATUS_ERROR

```
#define PNSO_ERR_CRYPTO_AXI_STATUS_ERROR 30004
```

4.1.1.19 PNSO_ERR_CRYPTO_GENERAL_ERROR

```
#define PNSO_ERR_CRYPTO_GENERAL_ERROR 30008
```

4.1.1.20 PNSO_ERR_CRYPTO_KEY_INDEX_OUT_OF_RANG

```
#define PNSO_ERR_CRYPTO_KEY_INDEX_OUT_OF_RANG 30001
```

4.1.1.21 PNSO_ERR_CRYPTO_KEY_NOT_REGISTERED

```
#define PNSO_ERR_CRYPTO_KEY_NOT_REGISTERED 30006
```

4.1.1.22 PNSO_ERR_CRYPTO_LEN_NOT_MULTI_SECTORS

```
#define PNSO_ERR_CRYPTO_LEN_NOT_MULTI_SECTORS 30007
```

4.1.1.23 PNSO_ERR_CRYPTO_WRONG_KEY_TYPE

```
#define PNSO_ERR_CRYPTO_WRONG_KEY_TYPE 30002
```

4.1.1.24 PNSO_ERR_SHA_FAILED

```
#define PNSO_ERR_SHA_FAILED 40001
```

4.1.1.25 PNSO_HASH_DFLAG_PER_BLOCK

```
#define PNSO_HASH_DFLAG_PER_BLOCK (1 << 0)
```

4.1.1.26 PNSO_HASH_TAG_LEN

```
#define PNSO_HASH_TAG_LEN 64
```

4.1.1.27 PNSO_MAX_HEADER_FIELDS

```
#define PNSO_MAX_HEADER_FIELDS 8
```

4.1.1.28 PNSO_OK

```
#define PNSO_OK 0
```

4.1.2 Typedef Documentation

4.1.2.1 completion_cb_t

```
typedef void(* completion_cb_t) (void *cb_ctx, struct pnso_service_result *svc_res)
```

typedef completion_cb_t: caller-supplied completion callback.

Parameters

cb_ctx	specifies the callback args for the caller-supplied callback routine.
svc_res	specifies a set of service results structures to report the status of each service within a request upon its completion.

4.1.2.2 pnso_error_t

```
typedef int32_t pnso_error_t
```

4.1.2.3 pnso_poll_fn_t

```
typedef pnso_error_t(* pnso_poll_fn_t) (void *pnso_poll_ctx)
```

typedef pnso_poll_fn_t: the caller to use this polling function to detect completion of a request.

Parameters

pnso_poll_ctx	[in] specifies the context passed as arg to the polling function. This context becomes invalid after exiting from completion callback.

Return Value: PNSO_OK - on success -EAGAIN - on request not done

4.1.3 Enumeration Type Documentation

4.1.3.1 pnso_chksum_type

`enum pnso_chksum_type`

Enumerator

PNSO_CHKSUM_TYPE_NONE	Algorithm for Checksum: None.
PNSO_CHKSUM_TYPE_MCRC64	Algorithm for Checksum: MCRC64.
PNSO_CHKSUM_TYPE_CRC32C	Algorithm for Checksum: CRC32C.
PNSO_CHKSUM_TYPE_ADLER32	Algorithm for Checksum: ADLER32.
PNSO_CHKSUM_TYPE_MADLER32	Algorithm for Checksum: MADLER32.
PNSO_CHKSUM_TYPE_MAX	NA.

4.1.3.2 pnso_compression_type

`enum pnso_compression_type`

Enumerator

PNSO_COMPRESSION_TYPE_NONE	Algorithm for Compression and Decompression: None.
PNSO_COMPRESSION_TYPE_LZRW1A	Algorithm for Compression and Decompression: LZRW1A.
PNSO_COMPRESSION_TYPE_MAX	NA.

4.1.3.3 pnso_crypto_type

`enum pnso_crypto_type`

Enumerator

PNSO_CRYPTO_TYPE_NONE	Algorithm for Encrption and Decryption: None.

Enumerator

PNSO_CRYPTO_TYPE_XTS	Algorithm for Encrption and Decryption: XTS.
PNSO_CRYPTO_TYPE_GCM	Algorithm for Encrption and Decryption: GCM.
PNSO_CRYPTO_TYPE_MAX	NA.

4.1.3.4 pnso_hash_type

`enum pnso_hash_type`

Enumerator

PNSO_HASH_TYPE_NONE	Algorithm for Deduplication hash: None.
PNSO_HASH_TYPE_SHA2_512	Algorithm for Deduplication hash: SHA2.
PNSO_HASH_TYPE_SHA2_256	Algorithm for Deduplication hash: SHA2.
PNSO_HASH_TYPE_MAX	NA.

4.1.3.5 pnso_header_field_type

`enum pnso_header_field_type`

enum pnso_header_field_type - defines the source for the compression header fields. PNSO_HDR_FIELD_TYP↩ E_STATIC - the field is used as input to set a fixed value in the compression header (ex: version).

PNSO_HDR_FIELD_TYPE_INDATA_CHKSUM - the field is used as input to set checksum value in the compression header derived from previous service.

PNSO_HDR_FIELD_TYPE_OUTDATA_LENGTH - the field is used as input to set length of the compressed buffer as the value in the compression header.

PNSO_HDR_FIELD_TYPE_ALGO - the field is used as input to set an algorithm type as the value in the compression header, taken from the hdr_algo field of the compression descriptor.

Enumerator

PNSO_HDR_FIELD_TYPE_NONE	Please see above for explanation.
PNSO_HDR_FIELD_TYPE_STATIC	Please see above for explanation.
PNSO_HDR_FIELD_TYPE_INDATA_CHKSUM	Please see above for explanation.
PNSO_HDR_FIELD_TYPE_OUTDATA_LENGTH	Please see above for explanation.
PNSO_HDR_FIELD_TYPE_ALGO	Please see above for explanation.
PNSO_HDR_FIELD_TYPE_MAX	

4.1.3.6 pnso_service_type

```
enum pnso_service_type
```

Pensando offloaders for storage and security

Pensando hardware accelerators can boost the storage and security performance.

By offloading compute-intensive workloads from the CPU core to Pensando hardware accelerators, performance and efficiency of the applications can significantly be increased.

This file contain declarations and functions that are necessary for applications to integrate Pensando hardware acceleration services.

Enumerator

PNSO_SVC_TYPE_NONE	Service: None.
PNSO_SVC_TYPE_ENCRYPT	Service: Encrypt.
PNSO_SVC_TYPE_DECRYPT	Service: Decrypt.
PNSO_SVC_TYPE_COMPRESS	Service: Compress.
PNSO_SVC_TYPE_DECOMPRESS	Service: Decompress.
PNSO_SVC_TYPE_HASH	Service: Hash.
PNSO_SVC_TYPE_CHKSUM	Service: Chksum.
PNSO_SVC_TYPE_DECOMPACT	Service: Decompact.
PNSO_SVC_TYPE_MAX	NA.

4.1.4 Function Documentation

4.1.4.1 pnso_add_compression_algo_mapping()

```
pnso_error_t pnso_add_compression_algo_mapping (
            enum pnso_compression_type pnso_algo,
            uint32_t header_algo )
```

pnso_add_compression_algo_mapping - Creates a mapping of Pensando compression algorithm number to algorithm number in compression header. Need to be done once during initialization.

Parameters

pnso_algo	[in] specifies the Pensando compression algorithm number.
header_algo	[in] specifies the compression header algorithm number.

Return Value: PNSO_OK - on success -EINVAL - on invalid input parameters

4.1.4.2 pnso_add_to_batch()

```
pnso_error_t pnso_add_to_batch (
```

```
            struct pnso_service_request * svc_req,
            struct pnso_service_result * svc_res )
```

pnso_add_to_batch() - routine that batches multiple requests and defers processing.

Parameters

svc_req	[in] specifies a set of service requests that to be used to complete the services within the request.
svc_res	[out] specifies a set of service results structures to report the status of each service within a request upon its completion.

Caller is responsible for allocation and deallocation of memory for both input and output parameters. Caller should keep the memory intact (ex: svc_req/svc_res) until the Pensando accelerator returns the result via completion callback.

None of the requests will be processed until the caller triggers a 'post' operation.

Even if any one of the request processing fails, the entire batch of requests will be dropped from further processing.

Return Value: PNSO_OK - on success -EINVAL - on invalid input parameters -ENOMEM - on failing to allocate memory

4.1.4.3 pnso_flush_batch()

```
pnso_error_t pnso_flush_batch (
            completion_cb_t cb,
            void * cb_ctx,
            pnso_poll_fn_t * pnso_poll_fn,
            void ** pnso_poll_ctx )
```

pnso_flush_batch() - routine that starts submitting the batched requests for further processing.

Parameters

cb	[in] specifies the caller-supplied completion callback routine.
cb_ctx	[in] specifies the caller-supplied context information.
pnso_poll_fn	[out] specifies the polling function, which the caller will use to poll for completion of the request.
pnso_poll_ctx	[out] specifies the context for the polling function.

Refer to 'pnso_service_result' and 'pnso_service_status' notes above for handling the output data.

Even if just the processing of flush request fails, the entire batch of requests will be dropped from further processing.

Return Value: PNSO_OK - on success -EINVAL - on invalid input parameters

4.1.4.4 pnso_init()

```
pnso_error_t pnso_init (
            struct pnso_init_params * init_params )
```

pnso_init() - initializes Pensando accelerators. Before using any of the Pensando accelerator services, this must be the first function to be invoked.

Parameters

init_params	[in] specifies the initialization parameters for Pensando Offloaders.

Caller is responsible for allocation and deallocation of memory for input parameters.

Return Value: PNSO_OK - on success -EINVAL - on invalid input parameters

4.1.4.5 pnso_register_compression_header_format()

```
pnso_error_t pnso_register_compression_header_format (
             struct pnso_compression_header_format * cp_hdr_fmt,
             uint16_t hdr_fmt_idx )
```

pnso_register_compression_header_format - Register a new header format. Needs to be done once during initialization.

Parameters

cp_hdr_fmt	[in] specified the header format to be embedded at beginning of compressed data.
hdr_fmt_idx	[in] Non-zero index to uniquely identify the header format.

Caller is responsible for managing the hdr_fmt_idx space and allocation/deallocation of memory for input parameters

Return Value: PNSO_OK - on success -EINVAL - on invalid input parameters

4.1.4.6 pnso_set_key_desc_idx()

```
pnso_error_t pnso_set_key_desc_idx (
             const void * key1,
             const void * key2,
             uint32_t key_size,
             uint32_t key_idx )
```

pnso_set_key_desc_idx() - sets the key descriptor index.

Parameters

key1	[in] specifies the key that will be used to encrypt the data.
key2	[in] specifies the key that will be used to encrypt initialization vector.
key_size	[in] specifies the size of the key in bytes – 16 and 32 bytes for AES128 and AES256 respectively.
key_idx	[in] specifies the key index in the descriptor table.

Caller is responsible for allocation and deallocation of memory for input parameters.

Return Value: PNSO_OK - on success -EINVAL - on invalid input parameters

4.1.4.7 pnso_submit_request()

```
pnso_error_t pnso_submit_request (
             struct pnso_service_request * svc_req,
             struct pnso_service_result * svc_res,
             completion_cb_t cb,
             void * cb_ctx,
             pnso_poll_fn_t * pnso_poll_fn,
             void ** pnso_poll_ctx )
```

pnso_submit_request() - routine that accepts one or more service(s) as a request and submits the request for further processing.

Parameters

svc_req	[in] specifies a set of service requests that to be used to complete the services within the request.
svc_res	[out] specifies a set of service results structures to report the status of each service within a request upon its completion.
cb	[in] specifies the caller-supplied completion callback routine.
cb_ctx	[in] specifies the caller-supplied context information.
pnso_poll_fn	[out] specifies the polling function, which the caller will use to poll for completion of the request.
pnso_poll_ctx	[out] specifies the context for the polling function.

Caller is responsible for allocation and deallocation of memory for both input and output parameters. Caller should keep the memory intact (ex: svc_req/svc_res) until the Pensando accelerator returns the result via completion callback.

Refer to 'pnso_service_result' and 'pnso_service_status' notes above for handling the output data.

Return Value: PNSO_OK - on success -EINVAL - on invalid input parameters -ENOMEM - on failing to allocate memory

Index