

PENSANDO

S Y S T E M S

Naples Offload API Reference Guide v. 0.1

Generated from SONIC driver version 1.0

Contents

- 1 Data Structure Index** **1**
- 1.1 Data Structures 1

- 2 File Index** **3**
- 2.1 File List 3

- 3 Data Structure Documentation** **5**
- 3.1 pnso_buffer_list Struct Reference 5
- 3.1.1 Detailed Description 5
- 3.1.2 Field Documentation 6
- 3.1.2.1 buffers 6
- 3.1.2.2 count 6
- 3.2 pnso_checksum_desc Struct Reference 6
- 3.2.1 Detailed Description 6
- 3.2.2 Field Documentation 7
- 3.2.2.1 algo_type 7
- 3.2.2.2 flags 7
- 3.3 pnso_chksum_tag Struct Reference 7
- 3.3.1 Detailed Description 7
- 3.3.2 Field Documentation 7
- 3.3.2.1 checksum 7
- 3.4 pnso_compression_desc Struct Reference 8
- 3.4.1 Detailed Description 8
- 3.4.2 Field Documentation 9

3.4.2.1	algo_type	9
3.4.2.2	flags	9
3.4.2.3	hdr_algo	9
3.4.2.4	hdr_fmt_idx	9
3.4.2.5	threshold_len	9
3.5	pnso_compression_header_format Struct Reference	10
3.5.1	Detailed Description	10
3.5.2	Field Documentation	10
3.5.2.1	fields	10
3.5.2.2	num_fields	11
3.6	pnso_crypto_desc Struct Reference	11
3.6.1	Detailed Description	11
3.6.2	Field Documentation	11
3.6.2.1	algo_type	11
3.6.2.2	iv_addr	12
3.6.2.3	key_desc_idx	12
3.6.2.4	rsvd	12
3.7	pnso_decompaction_desc Struct Reference	12
3.7.1	Detailed Description	12
3.7.2	Field Documentation	13
3.7.2.1	hdr_fmt_idx	13
3.7.2.2	rsvd_1	13
3.7.2.3	rsvd_2	13
3.7.2.4	vvbn	13
3.8	pnso_decompression_desc Struct Reference	13
3.8.1	Detailed Description	13
3.8.2	Field Documentation	14
3.8.2.1	algo_type	14
3.8.2.2	flags	14
3.8.2.3	hdr_fmt_idx	14

3.8.2.4	rsvd	14
3.9	pnso_flat_buffer Struct Reference	14
3.9.1	Detailed Description	14
3.9.2	Field Documentation	15
3.9.2.1	buf	15
3.9.2.2	len	15
3.10	pnso_hash_desc Struct Reference	15
3.10.1	Detailed Description	15
3.10.2	Field Documentation	16
3.10.2.1	algo_type	16
3.10.2.2	flags	16
3.11	pnso_hash_tag Struct Reference	16
3.11.1	Detailed Description	16
3.11.2	Field Documentation	16
3.11.2.1	hash	16
3.12	pnso_header_field Struct Reference	17
3.12.1	Detailed Description	17
3.12.2	Field Documentation	17
3.12.2.1	length	17
3.12.2.2	offset	17
3.12.2.3	type	18
3.12.2.4	value	18
3.13	pnso_init_params Struct Reference	18
3.13.1	Detailed Description	18
3.13.2	Field Documentation	18
3.13.2.1	block_size	18
3.13.2.2	per_core_qdepth	19
3.14	pnso_service Struct Reference	19
3.14.1	Detailed Description	19
3.14.2	Field Documentation	20

3.14.2.1	chksum_desc	20
3.14.2.2	cp_desc	20
3.14.2.3	crypto_desc	20
3.14.2.4	dc_desc	20
3.14.2.5	decompact_desc	20
3.14.2.6	hash_desc	21
3.14.2.7	rsvd	21
3.14.2.8	svc_type	21
3.14.2.9	u	21
3.15	pnso_service_request Struct Reference	21
3.15.1	Detailed Description	22
3.15.2	Field Documentation	22
3.15.2.1	num_services	22
3.15.2.2	sgl	22
3.15.2.3	svc	22
3.16	pnso_service_result Struct Reference	23
3.16.1	Detailed Description	23
3.16.2	Field Documentation	24
3.16.2.1	err	24
3.16.2.2	num_services	24
3.16.2.3	svc	24
3.17	pnso_service_status Struct Reference	24
3.17.1	Detailed Description	25
3.17.2	Field Documentation	25
3.17.2.1	chksum	25
3.17.2.2	data_len	26
3.17.2.3	dst	26
3.17.2.4	err	26
3.17.2.5	hash	26
3.17.2.6	num_tags	26
3.17.2.7	rsvd_1	26
3.17.2.8	rsvd_2	26
3.17.2.9	rsvd_3	26
3.17.2.10	sgl	27
3.17.2.11	svc_type	27
3.17.2.12	tags [1/2]	27
3.17.2.13	tags [2/2]	27
3.17.2.14	u	27

4 File Documentation	29
4.1 <code>pns0_api.h</code> File Reference	29
4.1.1 Macro Definition Documentation	31
4.1.1.1 <code>PNSO_CHKSUM_DFLAG_PER_BLOCK</code>	31
4.1.1.2 <code>PNSO_CHKSUM_TAG_LEN</code>	32
4.1.1.3 <code>PNSO_CP_DFLAG_BYPASS_ONFAIL</code>	32
4.1.1.4 <code>PNSO_CP_DFLAG_INSERT_HEADER</code>	32
4.1.1.5 <code>PNSO_CP_DFLAG_ZERO_PAD</code>	32
4.1.1.6 <code>PNSO_DC_DFLAG_HEADER_PRESENT</code>	32
4.1.1.7 <code>PNSO_ERR_CPDC_ALGO_INVALID</code>	32
4.1.1.8 <code>PNSO_ERR_CPDC_AXI_ADDR_ERROR</code>	32
4.1.1.9 <code>PNSO_ERR_CPDC_AXI_DATA_ERROR</code>	32
4.1.1.10 <code>PNSO_ERR_CPDC_AXI_TIMEOUT</code>	33
4.1.1.11 <code>PNSO_ERR_CPDC_CHECKSUM_FAILED</code>	33
4.1.1.12 <code>PNSO_ERR_CPDC_COMPRESSION_FAILED</code>	33
4.1.1.13 <code>PNSO_ERR_CPDC_DATA_TOO_LONG</code>	33
4.1.1.14 <code>PNSO_ERR_CPDC_HDR_IDX_INVALID</code>	33
4.1.1.15 <code>PNSO_ERR_CPDC_SGL_DESC_ERROR</code>	33
4.1.1.16 <code>PNSO_ERR_CRYPTO_AOL_DESC_ERROR</code>	33
4.1.1.17 <code>PNSO_ERR_CRYPTO_AXI_ERROR</code>	33
4.1.1.18 <code>PNSO_ERR_CRYPTO_AXI_STATUS_ERROR</code>	34
4.1.1.19 <code>PNSO_ERR_CRYPTO_GENERAL_ERROR</code>	34
4.1.1.20 <code>PNSO_ERR_CRYPTO_KEY_INDEX_OUT_OF_RANG</code>	34
4.1.1.21 <code>PNSO_ERR_CRYPTO_KEY_NOT_REGISTERED</code>	34
4.1.1.22 <code>PNSO_ERR_CRYPTO_LEN_NOT_MULTI_SECTORS</code>	34
4.1.1.23 <code>PNSO_ERR_CRYPTO_WRONG_KEY_TYPE</code>	34
4.1.1.24 <code>PNSO_ERR_SHA_FAILED</code>	34
4.1.1.25 <code>PNSO_HASH_DFLAG_PER_BLOCK</code>	34
4.1.1.26 <code>PNSO_HASH_TAG_LEN</code>	35
4.1.1.27 <code>PNSO_MAX_HEADER_FIELDS</code>	35

4.1.1.28	PNSO_OK	35
4.1.2	Typedef Documentation	35
4.1.2.1	completion_cb_t	35
4.1.2.2	pnso_error_t	35
4.1.2.3	pnso_poll_fn_t	35
4.1.3	Enumeration Type Documentation	36
4.1.3.1	pnso_chksum_type	36
4.1.3.2	pnso_compression_type	36
4.1.3.3	pnso_crypto_type	36
4.1.3.4	pnso_hash_type	37
4.1.3.5	pnso_header_field_type	37
4.1.3.6	pnso_service_type	38
4.1.4	Function Documentation	38
4.1.4.1	pnso_add_compression_algo_mapping()	38
4.1.4.2	pnso_add_to_batch()	38
4.1.4.3	pnso_flush_batch()	39
4.1.4.4	pnso_init()	39
4.1.4.5	pnso_register_compression_header_format()	40
4.1.4.6	pnso_set_key_desc_idx()	40
4.1.4.7	pnso_submit_request()	41
	Index	43

Chapter 1

Data Structure Index

1.1 Data Structures

Here are the data structures with brief descriptions:

- [pnso_buffer_list](#)
Describes a scatter/gather buffer list 5
- [pnso_checksum_desc](#)
Represents the descriptor for checksum operation 6
- [pnso_chksum_tag](#)
Represents the checksum tag 7
- [pnso_compression_desc](#)
Represents the descriptor for compression service 8
- [pnso_compression_header_format](#)
Represents the format of the compression header 10
- [pnso_crypto_desc](#)
Represents the descriptor for encryption or decryption operation 11
- [pnso_decompaction_desc](#)
Represents the descriptor for decompaction operation 12
- [pnso_decompression_desc](#)
Represents the descriptor for decompression operation 13
- [pnso_flat_buffer](#)
Describes a buffer with 'address and length' 14
- [pnso_hash_desc](#)
Represents the descriptor for data deduplication hash operation 15
- [pnso_hash_tag](#)
Represents the SHA hash tag 16
- [pnso_header_field](#)
Defines the value for each field in the compression header 17
- [pnso_init_params](#)
Represents the initialization parameters for Pensando accelerators 18
- [pnso_service](#)
Describes various parameters of the service chosen for acceleration 19
- [pnso_service_request](#)
Represents an array of services that are to be handled in a request 21
- [pnso_service_result](#)
Represents the result of the request upon completion one or all services 23
- [pnso_service_status](#)
Represents the result of a specific service within a request 24

Chapter 2

File Index

2.1 File List

Here is a list of all files with brief descriptions:

[pnso_api.h](#) 29

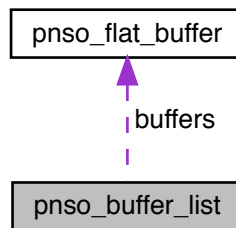
Chapter 3

Data Structure Documentation

3.1 pns0_buffer_list Struct Reference

Describes a scatter/gather buffer list.

Collaboration diagram for pns0_buffer_list:



Data Fields

- `uint32_t count`
- `struct pns0_flat_buffer buffers [0]`

3.1.1 Detailed Description

Describes a scatter/gather buffer list.

struct `pns0_buffer_list` - describes a scatter/gather buffer list.

Parameters

<i>count</i>	specifies the number of buffers in the list.
<i>buffers</i>	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 pns0_flat_buffer pns0_buffer_list::buffers[0]
```

3.1.2.2 count

```
uint32_t pns0_buffer_list::count
```

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

- [pns0_api.h](#)

3.2 pns0_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 [pns0_checksum_desc](#) - represents the descriptor for checksum operation.

Parameters

<i>algo_type</i>	specifies one of the enumerated values of the checksum algorithm (i.e. pns0_chksum_type).
<i>flags</i>	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 pns0_checksum_desc::algo_type
```

3.2.2.2 flags

```
uint16_t pns0_checksum_desc::flags
```

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

- [pns0_api.h](#)

3.3 pns0_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 [pns0_chksum_tag](#) - represents the checksum tag.

Parameters

<i>chksum</i>	specifies 4 or 8-byte checksum.
---------------	---------------------------------

3.3.2 Field Documentation

3.3.2.1 chksum

```
uint8_t pns0_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

<i>algo_type</i>	specifies one of the enumerated values of the compressor algorithm (i.e. pnso_compressor_type).
<i>flags</i>	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

<i>threshold_len</i>	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'.
<i>hdr_fmt_idx</i>	specifies the index for the header format in the header format array.
<i>hdr_algo</i>	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

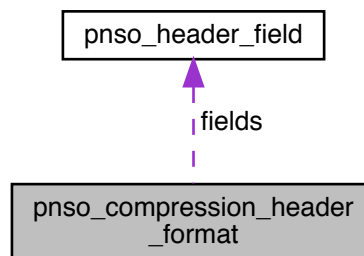
The documentation for this struct was generated from the following file:

- [pnso_api.h](#)

3.5 pns0_compression_header_format Struct Reference

Represents the format of the compression header.

Collaboration diagram for pns0_compression_header_format:



Data Fields

- `uint32_t num_fields`
- `struct pns0_header_field fields [PNSO_MAX_HEADER_FIELDS]`

3.5.1 Detailed Description

Represents the format of the compression header.

struct `pns0_compression_header_format` - represents the format of the compression header.

Parameters

<i>num_fields</i>	specifies the number of fields in the bounded array.
<i>fields</i>	specifies an array of fields.

3.5.2 Field Documentation

3.5.2.1 fields

```
struct pns0_header_field pns0_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

<i>algo_type</i>	specifies one of the enumerated values of the crypto type (i.e. pnso_crypto_type).
<i>rsvd</i>	specifies a 'reserved' field meant to be used by Pensando.
<i>key_desc_idx</i>	specifies the key index in the descriptor table.
<i>iv_addr</i>	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 pnsso_crypto_desc::iv_addr
```

3.6.2.3 key_desc_idx

```
uint32_t pnsso_crypto_desc::key_desc_idx
```

3.6.2.4 rsvd

```
uint16_t pnsso_crypto_desc::rsvd
```

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

- [pnsso_api.h](#)

3.7 pnsso_decompaction_desc Struct Reference

Represents the descriptor for decompaction operation.

Data Fields

- uint64_t [vbn](#):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 [pnsso_decompaction_desc](#) - represents the descriptor for decompaction operation.

Parameters

<i>vbn</i>	specifies the block number within the Netapp's packed block header, with which the offset and length of data can be retrieved.
<i>hdr_fmt_idx</i>	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

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

3.8.2 Field Documentation

3.8.2.1 algo_type

```
uint16_t pnsso_decompression_desc::algo_type
```

3.8.2.2 flags

```
uint16_t pnsso_decompression_desc::flags
```

3.8.2.3 hdr_fmt_idx

```
uint16_t pnsso_decompression_desc::hdr_fmt_idx
```

3.8.2.4 rsvd

```
uint16_t pnsso_decompression_desc::rsvd
```

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

- [pnsso_api.h](#)

3.9 pnsso_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 [pnsso_flat_buffer](#) - describes a buffer with 'address and length'.

Parameters

<i>len</i>	contains the length of the buffer in bytes.
<i>buf</i>	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

<i>algo_type</i>	specifies one of the enumerated values of the hash algorithm (i.e. <code>pnso_hash_type</code>) for data deduplication.
<i>flags</i>	specifies the following applicable descriptor flag(s) to hash descriptor. <code>PNSO_HASH_DFLAG_PER_BLOCK</code> - 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 pns0_hash_desc::algo_type
```

3.10.2.2 flags

```
uint16_t pns0_hash_desc::flags
```

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

- [pns0_api.h](#)

3.11 pns0_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 [pns0_hash_tag](#) - represents the SHA hash tag.

Parameters

<i>hash</i>	specifies a 64 or 32-byte hash.
-------------	---------------------------------

3.11.2 Field Documentation

3.11.2.1 hash

```
uint8_t pns0_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

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

<i>per_core_qdepth</i>	specifies the maximum number of parallel requests per core.
<i>block_size</i>	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
```

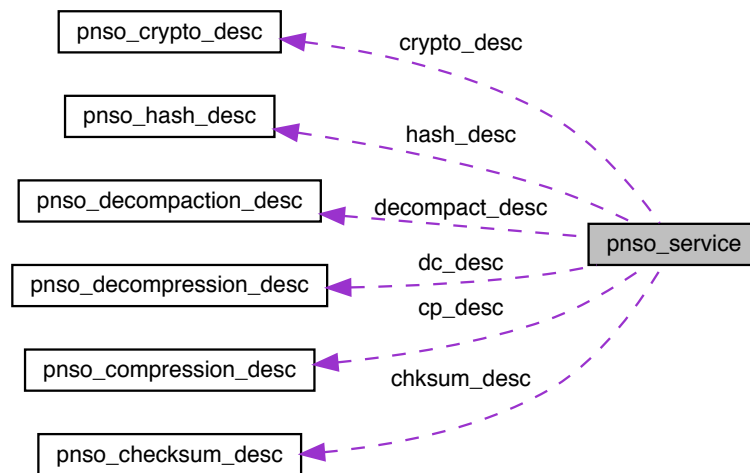
The documentation for this struct was generated from the following file:

- [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](#)

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

<i>svc_type</i>	specifies one of the enumerated values for the accelerator service type.
<i>crypto_desc</i>	specifies the descriptor for encryption/decryption service.
<i>cp_desc</i>	specifies the descriptor for compression service.
<i>dc_desc</i>	specifies the descriptor for decompression service.
<i>hash_desc</i>	specifies the descriptor for deduplication service.
<i>chksum_desc</i>	specifies the descriptor for checksum service.
<i>decompact_desc</i>	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 pnservice_hash_desc pnservice::hash_desc
```

3.14.2.7 rsvd

```
uint16_t pnservice::rsvd
```

3.14.2.8 svc_type

```
uint16_t pnservice::svc_type
```

3.14.2.9 u

```
union { ... } pnservice::u
```

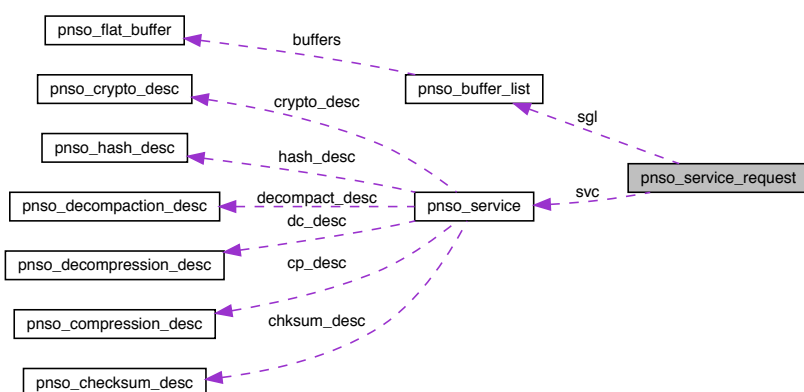
The documentation for this struct was generated from the following file:

- [pnservice_api.h](#)

3.15 pnservice_request Struct Reference

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

Collaboration diagram for pnservice_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

<i>sgl</i>	specifies input buffer list on which the request will operate on.
<i>num_services</i>	specifies the number of services in the input service request.
<i>svc</i>	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]
```

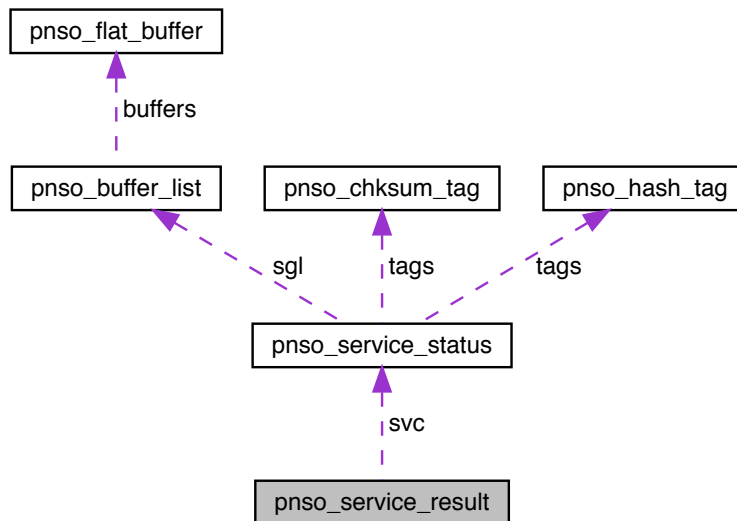
The documentation for this struct was generated from the following file:

- [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

<i>err</i>	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.
<i>num_services</i>	specifies the number of services in the request.
<i>svc</i>	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

```
pnsso_error_t pnsso_service_result::err
```

3.16.2.2 num_services

```
uint32_t pnsso_service_result::num_services
```

3.16.2.3 svc

```
struct pnsso_service_status pnsso_service_result::svc[0]
```

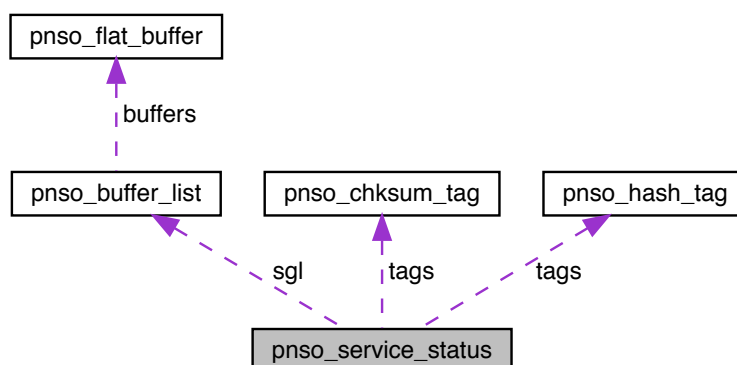
The documentation for this struct was generated from the following file:

- [pnsso_api.h](#)

3.17 pnsso_service_status Struct Reference

Represents the result of a specific service within a request.

Collaboration diagram for pnsso_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

<i>err</i>	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.
<i>svc_type</i>	specifies one of the enumerated values for the accelerator service type.
<i>rsvd_1</i>	specifies a 'reserved' field meant to be used by Pensando.
<i>hash</i>	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.
<i>chksum</i>	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.
<i>dst</i>	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 pnservice_status::data_len
```

3.17.2.3 dst

```
struct { ... } pnservice_status::dst
```

3.17.2.4 err

```
pnservice\_error\_t pnservice_status::err
```

3.17.2.5 hash

```
struct { ... } pnservice_status::hash
```

3.17.2.6 num_tags

```
uint16_t pnservice_status::num_tags
```

3.17.2.7 rsvd_1

```
uint16_t pnservice_status::rsvd_1
```

3.17.2.8 rsvd_2

```
uint16_t pnservice_status::rsvd_2
```

3.17.2.9 rsvd_3

```
uint16_t pnservice_status::rsvd_3
```

3.17.2.10 sgl

```
struct pnservice\_buffer\_list* pnservice_status::sgl
```

3.17.2.11 svc_type

```
uint16_t pnservice_status::svc_type
```

3.17.2.12 tags [1/2]

```
struct pnservice\_hash\_tag* pnservice_status::tags
```

3.17.2.13 tags [2/2]

```
struct pnservice\_chksum\_tag* pnservice_status::tags
```

3.17.2.14 u

```
union { ... } pnservice_status::u
```

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

- [pnservice_api.h](#)

Chapter 4

File Documentation

4.1 pns0_api.h File Reference

Data Structures

- struct [pns0_flat_buffer](#)
Describes a buffer with 'address and length'.
- struct [pns0_buffer_list](#)
Describes a scatter/gather buffer list.
- struct [pns0_header_field](#)
Defines the value for each field in the compression header.
- struct [pns0_compression_header_format](#)
Represents the format of the compression header.
- struct [pns0_init_params](#)
Represents the initialization parameters for Pensando accelerators.
- struct [pns0_crypto_desc](#)
Represents the descriptor for encryption or decryption operation.
- struct [pns0_compression_desc](#)
Represents the descriptor for compression service.
- struct [pns0_decompression_desc](#)
Represents the descriptor for decompression operation.
- struct [pns0_hash_desc](#)
Represents the descriptor for data deduplication hash operation.
- struct [pns0_checksum_desc](#)
Represents the descriptor for checksum operation.
- struct [pns0_decompaction_desc](#)
Represents the descriptor for decompaction operation.
- struct [pns0_hash_tag](#)
Represents the SHA hash tag.
- struct [pns0_chksum_tag](#)
Represents the checksum tag.
- struct [pns0_service_status](#)
Represents the result of a specific service within a request.
- struct [pns0_service_result](#)
Represents the result of the request upon completion one or all services.
- struct [pns0_service](#)
Describes various parameters of the service chosen for acceleration.
- struct [pns0_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_CRYPTOKEY_INDEX_OUT_OF_RANGE 30001`
- `#define PNSO_ERR_CRYPTOWRONG_KEY_TYPE 30002`
- `#define PNSO_ERR_CRYPTOAAXI_ERROR 30003`
- `#define PNSO_ERR_CRYPTOAAXI_STATUS_ERROR 30004`
- `#define PNSO_ERR_CRYPTOAOL_DESC_ERROR 30005`
- `#define PNSO_ERR_CRYPTOKEY_NOT_REGISTERED 30006`
- `#define PNSO_ERR_CRYPTOLEN_NOT_MULTI_SECTORS 30007`
- `#define PNSO_ERR_CRYPTOGENERAL_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_CRYPTOTYPE_NONE = 0,`
`PNSO_CRYPTOTYPE_XTS = 1,`
`PNSO_CRYPTOTYPE_GCM = 2,`
`PNSO_CRYPTOTYPE_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_chksm_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_CRYPT0_AOL_DESC_ERROR

```
#define PNSO_ERR_CRYPT0_AOL_DESC_ERROR 30005
```

4.1.1.17 PNSO_ERR_CRYPT0_AXI_ERROR

```
#define PNSO_ERR_CRYPT0_AXI_ERROR 30003
```

4.1.1.18 PNSO_ERR_CRYPT0_AXI_STATUS_ERROR

```
#define PNSO_ERR_CRYPT0_AXI_STATUS_ERROR 30004
```

4.1.1.19 PNSO_ERR_CRYPT0_GENERAL_ERROR

```
#define PNSO_ERR_CRYPT0_GENERAL_ERROR 30008
```

4.1.1.20 PNSO_ERR_CRYPT0_KEY_INDEX_OUT_OF_RANG

```
#define PNSO_ERR_CRYPT0_KEY_INDEX_OUT_OF_RANG 30001
```

4.1.1.21 PNSO_ERR_CRYPT0_KEY_NOT_REGISTERED

```
#define PNSO_ERR_CRYPT0_KEY_NOT_REGISTERED 30006
```

4.1.1.22 PNSO_ERR_CRYPT0_LEN_NOT_MULTI_SECTORS

```
#define PNSO_ERR_CRYPT0_LEN_NOT_MULTI_SECTORS 30007
```

4.1.1.23 PNSO_ERR_CRYPT0_WRONG_KEY_TYPE

```
#define PNSO_ERR_CRYPT0_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 pns0_service_result *svc_res)
```

typedef completion_cb_t: caller-supplied completion callback.

Parameters

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

4.1.2.2 pns0_error_t

```
typedef int32_t pns0_error_t
```

4.1.2.3 pns0_poll_fn_t

```
typedef pns0_error_t(* pns0_poll_fn_t) (void *pns0_poll_ctx)
```

typedef pns0_poll_fn_t: the caller to use this polling function to detect completion of a request.

Parameters

<code>pnso_poll_ctx</code>	[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

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

4.1.3.2 pnso_compression_type

enum `pnso_compression_type`

Enumerator

<code>PNSO_COMPRESSION_TYPE_NONE</code>	Algorithm for Compression and Decompression: None.
<code>PNSO_COMPRESSION_TYPE_LZRW1A</code>	Algorithm for Compression and Decompression: LZRW1A.
<code>PNSO_COMPRESSION_TYPE_MAX</code>	NA.

4.1.3.3 pnso_crypto_type

enum `pnso_crypto_type`

Enumerator

<code>PNSO_CRYPTOTYPE_NONE</code>	Algorithm for Encryption and Decryption: None.
-----------------------------------	--

Enumerator

PNSO_CRYPT0_TYPE_XTS	Algorithm for Encrption and Decryption: XTS.
PNSO_CRYPT0_TYPE_GCM	Algorithm for Encrption and Decryption: GCM.
PNSO_CRYPT0_TYPE_MAX	NA.

4.1.3.4 pns0_hash_type

enum [pns0_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 pns0_header_field_type

enum [pns0_header_field_type](#)

enum pns0_header_field_type - defines the source for the compression header fields. PNSO_HDR_FIELD_TYPE_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 pnservice_type

enum `pnservice_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: Decompack.
PNSO_SVC_TYPE_MAX	NA.

4.1.4 Function Documentation

4.1.4.1 pnservice_add_compression_algo_mapping()

```
pnservice_error_t pnservice_add_compression_algo_mapping (
    enum pnservice_compression_type pnservice_algo,
    uint32_t header_algo )
```

`pnservice_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

<code>pnservice_algo</code>	[in] specifies the Pensando compression algorithm number.
<code>header_algo</code>	[in] specifies the compression header algorithm number.

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

4.1.4.2 pnservice_add_to_batch()

```
pnservice_error_t pnservice_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

<i>svc_req</i>	[in] specifies a set of service requests that to be used to complete the services within the request.
<i>svc_res</i>	[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

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

<i>init_params</i>	[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 `pnsso_register_compression_header_format()`

```
pnsso_error_t pnsso_register_compression_header_format (
    struct pnsso_compression_header_format * cp_hdr_fmt,
    uint16_t hdr_fmt_idx )
```

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

Parameters

<i>cp_hdr_fmt</i>	[in] specified the header format to be embedded at beginning of compressed data.
<i>hdr_fmt_idx</i>	[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 `pnsso_set_key_desc_idx()`

```
pnsso_error_t pnsso_set_key_desc_idx (
    const void * key1,
    const void * key2,
    uint32_t key_size,
    uint32_t key_idx )
```

`pnsso_set_key_desc_idx()` - sets the key descriptor index.

Parameters

<i>key1</i>	[in] specifies the key that will be used to encrypt the data.
<i>key2</i>	[in] specifies the key that will be used to encrypt initialization vector.
<i>key_size</i>	[in] specifies the size of the key in bytes – 16 and 32 bytes for AES128 and AES256 respectively.
<i>key_idx</i>	[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 pns0_submit_request()

```

pns0_error_t pns0_submit_request (
    struct pns0_service_request * svc_req,
    struct pns0_service_result * svc_res,
    completion_cb_t cb,
    void * cb_ctx,
    pns0_poll_fn_t * pns0_poll_fn,
    void ** pns0_poll_ctx )

```

[pns0_submit_request\(\)](#) - routine that accepts one or more service(s) as a request and submits the request for further processing.

Parameters

<i>svc_req</i>	[in] specifies a set of service requests that to be used to complete the services within the request.
<i>svc_res</i>	[out] specifies a set of service results structures to report the status of each service within a request upon its completion.
<i>cb</i>	[in] specifies the caller-supplied completion callback routine.
<i>cb_ctx</i>	[in] specifies the caller-supplied context information.
<i>pns0_poll_fn</i>	[out] specifies the polling function, which the caller will use to poll for completion of the request.
<i>pns0_poll_ctx</i>	[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 '[pns0_service_result](#)' and '[pns0_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

- algo_type
 - [pnso_checksum_desc, 7](#)
 - [pnso_compression_desc, 9](#)
 - [pnso_crypto_desc, 11](#)
 - [pnso_decompression_desc, 14](#)
 - [pnso_hash_desc, 16](#)
- block_size
 - [pnso_init_params, 18](#)
- buf
 - [pnso_flat_buffer, 15](#)
- buffers
 - [pnso_buffer_list, 6](#)
- chksum
 - [pnso_chksum_tag, 7](#)
 - [pnso_service_status, 25](#)
- chksum_desc
 - [pnso_service, 20](#)
- completion_cb_t
 - [pnso_api.h, 35](#)
- count
 - [pnso_buffer_list, 6](#)
- cp_desc
 - [pnso_service, 20](#)
- crypto_desc
 - [pnso_service, 20](#)
- data_len
 - [pnso_service_status, 25](#)
- dc_desc
 - [pnso_service, 20](#)
- decompact_desc
 - [pnso_service, 20](#)
- dst
 - [pnso_service_status, 26](#)
- err
 - [pnso_service_result, 24](#)
 - [pnso_service_status, 26](#)
- fields
 - [pnso_compression_header_format, 10](#)
- flags
 - [pnso_checksum_desc, 7](#)
 - [pnso_compression_desc, 9](#)
 - [pnso_decompression_desc, 14](#)
 - [pnso_hash_desc, 16](#)
- hash
 - [pnso_hash_tag, 16](#)
 - [pnso_service_status, 26](#)
- hash_desc
 - [pnso_service, 20](#)
- hdr_algo
 - [pnso_compression_desc, 9](#)
- hdr_fmt_idx
 - [pnso_compression_desc, 9](#)
 - [pnso_decompaction_desc, 13](#)
 - [pnso_decompression_desc, 14](#)
- iv_addr
 - [pnso_crypto_desc, 11](#)
- key_desc_idx
 - [pnso_crypto_desc, 12](#)
- len
 - [pnso_flat_buffer, 15](#)
- length
 - [pnso_header_field, 17](#)
- num_fields
 - [pnso_compression_header_format, 10](#)
- num_services
 - [pnso_service_request, 22](#)
 - [pnso_service_result, 24](#)
- num_tags
 - [pnso_service_status, 26](#)
- offset
 - [pnso_header_field, 17](#)
- PNSO_CHKSUM_DFLAG_PER_BLOCK
 - [pnso_api.h, 31](#)
- PNSO_CHKSUM_TAG_LEN
 - [pnso_api.h, 31](#)
- PNSO_CP_DFLAG_BYPASS_ONFAIL
 - [pnso_api.h, 32](#)
- PNSO_CP_DFLAG_INSERT_HEADER
 - [pnso_api.h, 32](#)
- PNSO_CP_DFLAG_ZERO_PAD
 - [pnso_api.h, 32](#)
- PNSO_DC_DFLAG_HEADER_PRESENT
 - [pnso_api.h, 32](#)
- PNSO_ERR_CPDC_ALGO_INVALID
 - [pnso_api.h, 32](#)
- PNSO_ERR_CPDC_AXI_ADDR_ERROR
 - [pnso_api.h, 32](#)
- PNSO_ERR_CPDC_AXI_DATA_ERROR
 - [pnso_api.h, 32](#)
- PNSO_ERR_CPDC_AXI_TIMEOUT

- pnso_api.h, 32
- PNSO_ERR_CPDC_CHECKSUM_FAILED
 - pnso_api.h, 33
- PNSO_ERR_CPDC_COMPRESSION_FAILED
 - pnso_api.h, 33
- PNSO_ERR_CPDC_DATA_TOO_LONG
 - pnso_api.h, 33
- PNSO_ERR_CPDC_HDR_IDX_INVALID
 - pnso_api.h, 33
- PNSO_ERR_CPDC_SGL_DESC_ERROR
 - pnso_api.h, 33
- PNSO_ERR_CRYPTAO_AOL_DESC_ERROR
 - pnso_api.h, 33
- PNSO_ERR_CRYPTAO_AXI_ERROR
 - pnso_api.h, 33
- PNSO_ERR_CRYPTAO_AXI_STATUS_ERROR
 - pnso_api.h, 33
- PNSO_ERR_CRYPTAO_GENERAL_ERROR
 - pnso_api.h, 34
- PNSO_ERR_CRYPTAO_KEY_INDEX_OUT_OF_RANG
 - pnso_api.h, 34
- PNSO_ERR_CRYPTAO_KEY_NOT_REGISTERED
 - pnso_api.h, 34
- PNSO_ERR_CRYPTAO_LEN_NOT_MULTI_SECTORS
 - pnso_api.h, 34
- PNSO_ERR_CRYPTAO_WRONG_KEY_TYPE
 - pnso_api.h, 34
- PNSO_ERR_SHA_FAILED
 - pnso_api.h, 34
- PNSO_HASH_DFLAG_PER_BLOCK
 - pnso_api.h, 34
- PNSO_HASH_TAG_LEN
 - pnso_api.h, 34
- PNSO_MAX_HEADER_FIELDS
 - pnso_api.h, 35
- PNSO_OK
 - pnso_api.h, 35
- per_core_qdepth
 - pnso_init_params, 18
- pnso_add_compression_algo_mapping
 - pnso_api.h, 38
- pnso_add_to_batch
 - pnso_api.h, 38
- pnso_api.h, 29
 - completion_cb_t, 35
 - PNSO_CHKSUM_DFLAG_PER_BLOCK, 31
 - PNSO_CHKSUM_TAG_LEN, 31
 - PNSO_CP_DFLAG_BYPASS_ONFAIL, 32
 - PNSO_CP_DFLAG_INSERT_HEADER, 32
 - PNSO_CP_DFLAG_ZERO_PAD, 32
 - PNSO_DC_DFLAG_HEADER_PRESENT, 32
 - PNSO_ERR_CPDC_ALGO_INVALID, 32
 - PNSO_ERR_CPDC_AXI_ADDR_ERROR, 32
 - PNSO_ERR_CPDC_AXI_DATA_ERROR, 32
 - PNSO_ERR_CPDC_AXI_TIMEOUT, 32
 - PNSO_ERR_CPDC_CHECKSUM_FAILED, 33
 - PNSO_ERR_CPDC_COMPRESSION_FAILED, 33
- PNSO_ERR_CPDC_DATA_TOO_LONG, 33
- PNSO_ERR_CPDC_HDR_IDX_INVALID, 33
- PNSO_ERR_CPDC_SGL_DESC_ERROR, 33
- PNSO_ERR_CRYPTAO_AOL_DESC_ERROR, 33
- PNSO_ERR_CRYPTAO_AXI_ERROR, 33
- PNSO_ERR_CRYPTAO_AXI_STATUS_ERROR, 33
- PNSO_ERR_CRYPTAO_GENERAL_ERROR, 34
- PNSO_ERR_CRYPTAO_KEY_INDEX_OUT_OF_RANG, 34
- PNSO_ERR_CRYPTAO_KEY_NOT_REGISTERED, 34
- PNSO_ERR_CRYPTAO_LEN_NOT_MULTI_SECTORS, 34
- PNSO_ERR_CRYPTAO_WRONG_KEY_TYPE, 34
- PNSO_ERR_SHA_FAILED, 34
- PNSO_HASH_DFLAG_PER_BLOCK, 34
- PNSO_HASH_TAG_LEN, 34
- PNSO_MAX_HEADER_FIELDS, 35
- PNSO_OK, 35
- pnso_add_compression_algo_mapping, 38
- pnso_add_to_batch, 38
- pnso_chksum_type, 36
- pnso_compression_type, 36
- pnso_crypto_type, 36
- pnso_error_t, 35
- pnso_flush_batch, 39
- pnso_hash_type, 37
- pnso_header_field_type, 37
- pnso_init, 39
- pnso_poll_fn_t, 35
- pnso_register_compression_header_format, 40
- pnso_service_type, 37
- pnso_set_key_desc_idx, 40
- pnso_submit_request, 40
- pnso_buffer_list, 5
 - buffers, 6
 - count, 6
- pnso_checksum_desc, 6
 - algo_type, 7
 - flags, 7
- pnso_chksum_tag, 7
 - chksum, 7
- pnso_chksum_type
 - pnso_api.h, 36
- pnso_compression_desc, 8
 - algo_type, 9
 - flags, 9
 - hdr_algo, 9
 - hdr_fmt_idx, 9
 - threshold_len, 9
- pnso_compression_header_format, 10
 - fields, 10
 - num_fields, 10
- pnso_compression_type
 - pnso_api.h, 36
- pnso_crypto_desc, 11
 - algo_type, 11

- iv_addr, 11
- key_desc_idx, 12
- rsvd, 12
- pnso_crypto_type
 - pnso_api.h, 36
- pnso_decompaction_desc, 12
 - hdr_fmt_idx, 13
 - rsvd_1, 13
 - rsvd_2, 13
 - vbn, 13
- pnso_decompression_desc, 13
 - algo_type, 14
 - flags, 14
 - hdr_fmt_idx, 14
 - rsvd, 14
- pnso_error_t
 - pnso_api.h, 35
- pnso_flat_buffer, 14
 - buf, 15
 - len, 15
- pnso_flush_batch
 - pnso_api.h, 39
- pnso_hash_desc, 15
 - algo_type, 16
 - flags, 16
- pnso_hash_tag, 16
 - hash, 16
- pnso_hash_type
 - pnso_api.h, 37
- pnso_header_field, 17
 - length, 17
 - offset, 17
 - type, 17
 - value, 18
- pnso_header_field_type
 - pnso_api.h, 37
- pnso_init
 - pnso_api.h, 39
- pnso_init_params, 18
 - block_size, 18
 - per_core_qdepth, 18
- pnso_poll_fn_t
 - pnso_api.h, 35
- pnso_register_compression_header_format
 - pnso_api.h, 40
- pnso_service, 19
 - chksum_desc, 20
 - cp_desc, 20
 - crypto_desc, 20
 - dc_desc, 20
 - decompact_desc, 20
 - hash_desc, 20
 - rsvd, 21
 - svc_type, 21
 - u, 21
- pnso_service_request, 21
 - num_services, 22
 - sgl, 22
 - svc, 22
- pnso_service_result, 23
 - err, 24
 - num_services, 24
 - svc, 24
- pnso_service_status, 24
 - chksum, 25
 - data_len, 25
 - dst, 26
 - err, 26
 - hash, 26
 - num_tags, 26
 - rsvd_1, 26
 - rsvd_2, 26
 - rsvd_3, 26
 - sgl, 26
 - svc_type, 27
 - tags, 27
 - u, 27
- pnso_service_type
 - pnso_api.h, 37
- pnso_set_key_desc_idx
 - pnso_api.h, 40
- pnso_submit_request
 - pnso_api.h, 40
- rsvd
 - pnso_crypto_desc, 12
 - pnso_decompression_desc, 14
 - pnso_service, 21
- rsvd_1
 - pnso_decompaction_desc, 13
 - pnso_service_status, 26
- rsvd_2
 - pnso_decompaction_desc, 13
 - pnso_service_status, 26
- rsvd_3
 - pnso_service_status, 26
- sgl
 - pnso_service_request, 22
 - pnso_service_status, 26
- svc
 - pnso_service_request, 22
 - pnso_service_result, 24
- svc_type
 - pnso_service, 21
 - pnso_service_status, 27
- tags
 - pnso_service_status, 27
- threshold_len
 - pnso_compression_desc, 9
- type
 - pnso_header_field, 17
- u
 - pnso_service, 21
 - pnso_service_status, 27

value

pnso_header_field, [18](#)

vbn

pnso_decompaction_desc, [13](#)