Intel® QuickAssist Technology Cryptographic API Reference

Automatically generated from sources, Thu Apr 7 2022.

Reference Number: 330685-009

0.1 Revision History

Date	Revision	Description
April 2022	009	Changed version of the crypto API to v3.0. Added RSA8K support.
May 2021	008	Changed version of the crypto API to v2.5. Added support for SM2.
Nov 2020	007	Changed version of the crypto API to v2.4. Added support for ChaCha20-Poly1305. Added support for SM4 in ECB, CBC and CTR modes. Added support for SM3. Added support for SHA3-224, SHA3-384 and SHA3-512.
March 2020	006	Added HKDF API Added 22519 and 448 curve support to cpa_cy_ec.h
April 2018	005	Added session update API
July 2016	004	Added Intel Key Protection Technology (KPT) API
October 2015	003	Changed version of the crypto API for v2.0. Added ZUC-EEA3 and ZUC-EIA3 support Added SHA3-256 support
Sept 2015	002	Incrementing crypto API versio to v1.9. Adding CPA_STATUS_UNSUPPORTED as a return status.
June 2014	001	First "public" version of the document. Based on "Intel Confidential" document number 410923-1.8.

Chapter 1

Deprecated List

Global CPA_DEPRECATED

As of v1.3 of the Crypto API, this structure has been deprecated, replaced by CpaCyDhStats64.

Global CPA_DEPRECATED

As of v1.3 of the Crypto API, this structure has been deprecated, replaced by CpaCyDsaStats64.

Global CPA_DEPRECATED

As of v1.3 of the Crypto API, this structure has been deprecated, replaced by CpaCyLnStats64.

Global CPA_DEPRECATED

As of v1.3 of the Crypto API, this structure has been deprecated, replaced by CpaInstanceInfo2.

Global CPA_DEPRECATED

As of v1.3 of the Crypto API, this enum has been deprecated, replaced by CpaAccelerationServiceType.

As of v1.3 of the Crypto API, this enum has been deprecated, replaced by CpaOperationalState.

As of v1.3 of the Crypto API, this structure has been deprecated, replaced by CpaInstanceInfo2.

As of v1.3 of the cryptographic API, this structure has been deprecated, replaced by CpaCySymStats64.

As of v1.3 of the Crypto API, this structure has been deprecated, replaced by CpaCyKeyGenStats64.

As of v1.3 of the Crypto API, this structure has been deprecated, replaced by CpaCyRsaStats64.

As of v1.3 of the Crypto API, this structure has been deprecated, replaced by CpaCyDhStats64.

As of v1.3 of the Crypto API, this structure has been deprecated, replaced by CpaCyDsaStats64.

As of v1.3 of the Crypto API, this structure has been deprecated, replaced by CpaCyLnStats64.

As of v1.3 of the Crypto API, this structure has been deprecated, replaced by CpaCyPrimeStats64.

Global CPA_DEPRECATED

As of v1.3 of the Crypto API, this structure has been deprecated, replaced by CpaCyPrimeStats64.

Global CPA_DEPRECATED

As of v1.3 of the Crypto API, this enum has been deprecated, replaced by CpaOperationalState.

Global CPA_DEPRECATED

As of v1.3 of the Crypto API, this enum has been deprecated, replaced by CpaAccelerationServiceType.

As of v1.3 of the Crypto API, this enum has been deprecated, replaced by CpaOperationalState.

As of v1.3 of the Crypto API, this structure has been deprecated, replaced by CpaInstanceInfo2.

As of v1.3 of the cryptographic API, this structure has been deprecated, replaced by CpaCySymStats64.

As of v1.3 of the Crypto API, this structure has been deprecated, replaced by CpaCyKeyGenStats64.

As of v1.3 of the Crypto API, this structure has been deprecated, replaced by CpaCyRsaStats64.

As of v1.3 of the Crypto API, this structure has been deprecated, replaced by CpaCyDhStats64.

As of v1.3 of the Crypto API, this structure has been deprecated, replaced by CpaCyDsaStats64.

As of v1.3 of the Crypto API, this structure has been deprecated, replaced by CpaCyLnStats64.

As of v1.3 of the Crypto API, this structure has been deprecated, replaced by CpaCyPrimeStats64.

Global CPA_DEPRECATED

As of v1.3 of the Crypto API, this structure has been deprecated, replaced by CpaCyKeyGenStats64.

Global CPA_DEPRECATED

As of v1.3 of the Crypto API, this structure has been deprecated, replaced by CpaCyRsaStats64.

Global CPA_DEPRECATED

As of v1.3 of the cryptographic API, this structure has been deprecated, replaced by CpaCySymStats64.

Global cpaCyDhQueryStats (const CpaInstanceHandle instanceHandle, struct _CpaCyDhStats ∗pDhStats)

As of v1.3 of the Crypto API, this function has been deprecated, replaced by cpaCyDhQueryStats64().

Global cpaCyDsaQueryStats (const CpaInstanceHandle instanceHandle, struct _CpaCyDsaStats ∗pDsa←↩Stats)

As of v1.3 of the Crypto API, this function has been deprecated, replaced by cpaCyDsaQueryStats64().

Global cpaCyEcPointMultiply (const CpaInstanceHandle instanceHandle, const CpaCyEcPointMultiply←↩CbFunc pCb, void ∗pCallbackTag, const CpaCyEcPointMultiplyOpData ∗pOpData, CpaBoolean ∗p←↩MultiplyStatus, CpaFlatBuffer ∗pXk, CpaFlatBuffer ∗pYk)

This function is replaced with cpaCyEcGenericPointMultiply

Global cpaCyEcPointVerify (const CpaInstanceHandle instanceHandle, const CpaCyEcPointVerifyCbFunc pCb, void ∗pCallbackTag, const CpaCyEcPointVerifyOpData ∗pOpData, CpaBoolean ∗pVerifyStatus)

This function is replaced with cpaCyEcGenericPointVerify

Global cpaCyInstanceGetInfo (const CpaInstanceHandle instanceHandle, struct _CpaInstanceInfo ∗p←↩InstanceInfo)

As of v1.3 of the Crypto API, this function has been deprecated, replaced by cpaCyInstanceGetInfo2.

Global cpaCyKeyGenQueryStats (const CpaInstanceHandle instanceHandle, struct _CpaCyKeyGenStats ∗pKeyGenStats)

As of v1.3 of the Crypto API, this function has been deprecated, replaced by cpaCyKeyGenQueryStats64().

Global cpaCyLnStatsQuery (const CpaInstanceHandle instanceHandle, struct _CpaCyLnStats ∗pLnStats)

As of v1.3 of the Crypto API, this function has been deprecated, replaced by cpaCyLnStatsQuery64().

Global cpaCyRsaQueryStats (const CpaInstanceHandle instanceHandle, struct _CpaCyRsaStats ∗pRsa←↩Stats)

As of v1.3 of the Crypto API, this function has been deprecated, replaced by cpaCyRsaQueryStats64().

Global cpaCySymQueryStats (const CpaInstanceHandle instanceHandle, struct _CpaCySymStats ∗pSym←↩Stats)

As of v1.3 of the cryptographic API, this function has been deprecated, replaced by cpaCySymQueryStats64().

Chapter 2

Module Index

2.1 Modules

Here is a list of all modules:

Chapter 3

Data Structure Index

3.1 Data Structures

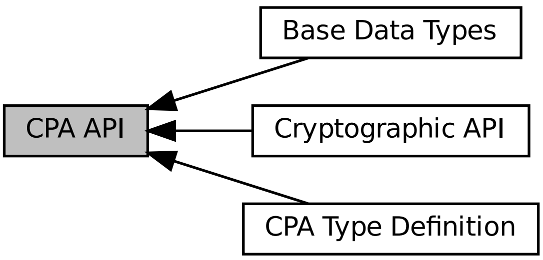
Here are the data structures with brief descriptions:

Chapter 4

Module Documentation

4.1 CPA API

Collaboration diagram for CPA API:



Modules

- Base Data Types
- CPA Type Definition
- Cryptographic API

Functions

- CpaStatus cpaGetNumInstances (const CpaAccelerationServiceType accelerationServiceType, Cpa16U ∗pNumInstances)
- CpaStatus cpaGetInstances (const CpaAccelerationServiceType accelerationServiceType, Cpa16U numInstances, CpaInstanceHandle ∗cpaInstances)

4.1.1 Detailed Description

File: cpa.h

Description:

This is the top level API definition for Intel(R) QuickAssist Technology. It contains structures, data types and definitions that are common across the interface.

4.1.2 Function Documentation

4.1.2.1 cpaGetNumInstances()

```
CpaStatus cpaGetNumInstances (
            const CpaAccelerationServiceType accelerationServiceType,
            Cpa16U * pNumInstances )
```

File: cpa.h

Get the number of Acceleration Service instances that are supported by the API implementation.

Description:

This function will get the number of instances that are supported for the required Acceleration Service by an implementation of the CPA API. This number is then used to determine the size of the array that must be passed to cpaGetInstances().

Context:

This function MUST NOT be called from an interrupt context as it MAY sleep.

Assumptions:

None

Side-Effects:

None

Blocking:

This function is synchronous and blocking.

Reentrant:

No

Thread-safe:

Yes

Parameters

in	*accelerationServiceType*	Acceleration Service required
out	*pNumInstances*	Pointer to where the number of instances will be written.

Return values

CPA_STATUS_SUCCESS	Function executed successfully.
CPA_STATUS_FAIL	Function failed.
CPA_STATUS_INVALID_PARAM	Invalid parameter passed in.
CPA_STATUS_UNSUPPORTED	Function is not supported.

Precondition

None

Postcondition

None

Note

This function operates in a synchronous manner and no asynchronous callback will be generated

See also

cpaGetInstances

4.1.2.2 cpaGetInstances()

```
CpaStatus cpaGetInstances (
            const CpaAccelerationServiceType accelerationServiceType,
            Cpa16U numInstances,
            CpaInstanceHandle * cpaInstances )
```

File: cpa.h

Get the handles to the required Acceleration Service instances that are supported by the API implementation.

Description:

This function will return handles to the required Acceleration Service instances that are supported by an implementation of the CPA API. These instance handles can then be used as input parameters with other API functions.

This function will populate an array that has been allocated by the caller. The size of this array will have been determined by the cpaGetNumInstances() function.

Context:

This function MUST NOT be called from an interrupt context as it MAY sleep.

Assumptions:

None

Side-Effects:

None

Blocking:

This function is synchronous and blocking.

Reentrant:

No

Thread-safe:

Yes

Parameters

in	*accelerationServiceType*	Acceleration Service requested
in	*numInstances*	Size of the array. If the value is greater than the number of instances supported, then an error (CPA_STATUS_INVALID_PARAM) is returned.
in,out	*cpaInstances*	Pointer to where the instance handles will be written.

Return values

CPA_STATUS_SUCCESS	Function executed successfully.
CPA_STATUS_FAIL	Function failed.
CPA_STATUS_INVALID_PARAM	Invalid parameter passed in.
CPA_STATUS_UNSUPPORTED	Function is not supported.

Precondition

None

Postcondition

None

Note

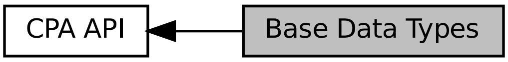
This function operates in a synchronous manner and no asynchronous callback will be generated

See also

cpaGetNumInstances

4.2 Base Data Types

Collaboration diagram for Base Data Types:



Data Structures

- struct _CpaFlatBuffer
- struct _CpaBufferList
- struct _CpaPhysFlatBuffer
- struct _CpaPhysBufferList
- struct _CpaInstanceInfo
- struct _CpaPhysicalInstanceId
- struct _CpaInstanceInfo2

Macros

- #define CPA_INSTANCE_HANDLE_SINGLE
- #define CPA_DP_BUFLIST
- #define CPA_STATUS_SUCCESS
- #define CPA_STATUS_FAIL
- #define CPA_STATUS_RETRY
- #define CPA_STATUS_RESOURCE
- #define CPA_STATUS_INVALID_PARAM
- #define CPA_STATUS_FATAL
- #define CPA_STATUS_UNSUPPORTED
- #define CPA_STATUS_RESTARTING
- #define CPA_STATUS_MAX_STR_LENGTH_IN_BYTES
- #define CPA_STATUS_STR_SUCCESS
- #define CPA_STATUS_STR_FAIL
- #define CPA_STATUS_STR_RETRY
- #define CPA_STATUS_STR_RESOURCE
- #define CPA_STATUS_STR_INVALID_PARAM
- #define CPA_STATUS_STR_FATAL
- #define CPA_STATUS_STR_UNSUPPORTED
- #define CPA_INSTANCE_MAX_NAME_SIZE_IN_BYTES
- #define CPA_INSTANCE_MAX_ID_SIZE_IN_BYTES
- #define CPA_INSTANCE_MAX_VERSION_SIZE_IN_BYTES

Typedefs

- typedef void * CpaInstanceHandle
- typedef Cpa64U CpaPhysicalAddr
- typedef CpaPhysicalAddr(* CpaVirtualToPhysical) (void *pVirtualAddr)
- typedef struct _CpaFlatBuffer CpaFlatBuffer
- typedef struct _CpaBufferList CpaBufferList
- typedef struct _CpaPhysFlatBuffer CpaPhysFlatBuffer
- typedef struct _CpaPhysBufferList CpaPhysBufferList
- typedef Cpa32S CpaStatus
- typedef enum _CpaInstanceType CPA_DEPRECATED
- typedef enum _CpaAccelerationServiceType CpaAccelerationServiceType
- typedef enum _CpaOperationalState CpaOperationalState
- typedef struct _CpaPhysicalInstanceId CpaPhysicalInstanceId
- typedef struct _CpaInstanceInfo2 CpaInstanceInfo2
- typedef enum _CpaInstanceEvent CpaInstanceEvent

Enumerations

- enum _CpaInstanceType
- enum _CpaAccelerationServiceType
- enum _CpaInstanceState
- enum _CpaOperationalState
- enum _CpaInstanceEvent

4.2.1 Detailed Description

File: cpa.h

Description:

The base data types for the Intel CPA API.

4.2.2 Macro Definition Documentation

4.2.2.1 CPA_INSTANCE_HANDLE_SINGLE

```
#define CPA_INSTANCE_HANDLE_SINGLE
```

Default instantiation handle value where there is only a single instance

Description:

Used as an instance handle value where only one instance exists.

Definition at line 70 of file cpa.h.

4.2.2.2 CPA_DP_BUFLIST

```
#define CPA_DP_BUFLIST
```

Special value which can be taken by length fields on some of the "data plane" APIs to indicate that the buffer in question is of type CpaPhysBufferList, rather than simply an array of bytes.

Definition at line 243 of file cpa.h.

4.2.2.3 CPA_STATUS_SUCCESS

```
#define CPA_STATUS_SUCCESS
```

Success status value.

Definition at line 258 of file cpa.h.

4.2.2.4 CPA_STATUS_FAIL

```
#define CPA_STATUS_FAIL
```

Fail status value.

Definition at line 262 of file cpa.h.

4.2.2.5 CPA_STATUS_RETRY

```
#define CPA_STATUS_RETRY
```

Retry status value.

Definition at line 266 of file cpa.h.

4.2.2.6 CPA_STATUS_RESOURCE

```
#define CPA_STATUS_RESOURCE
```

The resource that has been requested is unavailable. Refer to relevant sections of the API for specifics on what the suggested course of action is.

Definition at line 270 of file cpa.h.

4.2.2.7 CPA_STATUS_INVALID_PARAM

```
#define CPA_STATUS_INVALID_PARAM
```

Invalid parameter has been passed in.

Definition at line 276 of file cpa.h.

4.2.2.8 CPA_STATUS_FATAL

```
#define CPA_STATUS_FATAL
```

A serious error has occurred. Recommended course of action is to shutdown and restart the component.

Definition at line 280 of file cpa.h.

4.2.2.9 CPA_STATUS_UNSUPPORTED

```
#define CPA_STATUS_UNSUPPORTED
```

The function is not supported, at least not with the specific parameters supplied. This may be because a particular capability is not supported by the current implementation.

Definition at line 285 of file cpa.h.

4.2.2.10 CPA_STATUS_RESTARTING

```
#define CPA_STATUS_RESTARTING
```

The API implementation is restarting. This may be reported if, for example, a hardware implementation is undergoing a reset. Recommended course of action is to retry the request.

Definition at line 291 of file cpa.h.

4.2.2.11 CPA_STATUS_MAX_STR_LENGTH_IN_BYTES

```
#define CPA_STATUS_MAX_STR_LENGTH_IN_BYTES
```

API status string type definition

Description:

> This type definition is used for the generic status text strings provided by cpaXxGetStatusText API functions. Common values are defined, for example see CPA_STATUS_STR_SUCCESS, CPA_STATUS_FAIL, etc., as well as the maximum size CPA_STATUS_MAX_STR_LENGTH_IN_BYTES.

Maximum length of the Overall Status String (including generic and specific strings returned by calls to cpaXxGetStatusText)

Definition at line 309 of file cpa.h.

4.2.2.12 CPA_STATUS_STR_SUCCESS

```
#define CPA_STATUS_STR_SUCCESS
```

Status string for CPA_STATUS_SUCCESS.

Definition at line 315 of file cpa.h.

4.2.2.13 CPA_STATUS_STR_FAIL

```
#define CPA_STATUS_STR_FAIL
```

Status string for CPA_STATUS_FAIL.

Definition at line 319 of file cpa.h.

4.2.2.14 CPA_STATUS_STR_RETRY

```
#define CPA_STATUS_STR_RETRY
```

Status string for CPA_STATUS_RETRY.

Definition at line 323 of file cpa.h.

4.2.2.15 CPA_STATUS_STR_RESOURCE

```
#define CPA_STATUS_STR_RESOURCE
```

Status string for CPA_STATUS_RESOURCE.

Definition at line 327 of file cpa.h.

4.2.2.16 CPA_STATUS_STR_INVALID_PARAM

```
#define CPA_STATUS_STR_INVALID_PARAM
```

Status string for CPA_STATUS_INVALID_PARAM.

Definition at line 331 of file cpa.h.

4.2.2.17 CPA_STATUS_STR_FATAL

```
#define CPA_STATUS_STR_FATAL
```

Status string for CPA_STATUS_FATAL.

Definition at line 335 of file cpa.h.

4.2.2.18 CPA_STATUS_STR_UNSUPPORTED

```
#define CPA_STATUS_STR_UNSUPPORTED
```

Status string for CPA_STATUS_UNSUPPORTED.

Definition at line 339 of file cpa.h.

4.2.2.19 CPA_INSTANCE_MAX_NAME_SIZE_IN_BYTES

```
#define CPA_INSTANCE_MAX_NAME_SIZE_IN_BYTES
```

Maximum instance info name string length in bytes

Definition at line 437 of file cpa.h.

4.2.2.20 CPA_INSTANCE_MAX_ID_SIZE_IN_BYTES

```
#define CPA_INSTANCE_MAX_ID_SIZE_IN_BYTES
```

Maximum instance info id string length in bytes

Definition at line 441 of file cpa.h.

4.2.2.21 CPA_INSTANCE_MAX_VERSION_SIZE_IN_BYTES

```
#define CPA_INSTANCE_MAX_VERSION_SIZE_IN_BYTES
```

Maximum instance info version string length in bytes

Definition at line 445 of file cpa.h.

4.2.3 Typedef Documentation

4.2.3.1 CpaInstanceHandle

```
typedef void* CpaInstanceHandle
```

Instance handle type.

Description:

Handle used to uniquely identify an instance.

Note

Where only a single instantiation exists this field may be set to CPA_INSTANCE_HANDLE_SINGLE.

Definition at line 59 of file cpa.h.

4.2.3.2 CpaPhysicalAddr

```
typedef Cpa64U CpaPhysicalAddr
```

Physical memory address.

Description:

Type for physical memory addresses.

Definition at line 79 of file cpa.h.

4.2.3.3 CpaVirtualToPhysical

```
typedef CpaPhysicalAddr(* CpaVirtualToPhysical) (void *pVirtualAddr)
```

Virtual to physical address conversion routine.

Description:

This function is used to convert virtual addresses to physical addresses.

Context:

The function shall not be called in an interrupt context.

Assumptions:

None

Side-Effects:

None

Blocking:

This function is synchronous and blocking.

Reentrant:

No

Thread-safe:

Yes

Parameters

in	*pVirtualAddr*	Virtual address to be converted.

Returns

Returns the corresponding physical address. On error, the value NULL is returned.

Postcondition

None

See also

None

Definition at line 115 of file cpa.h.

4.2.3.4 CpaFlatBuffer

```
typedef struct _CpaFlatBuffer CpaFlatBuffer
```

Flat buffer structure containing a pointer and length member.

Description:

A flat buffer structure. The data pointer, pData, is a virtual address. An API instance may require the actual data to be in contiguous physical memory as determined by CpaInstanceInfo2.

4.2.3.5 CpaBufferList

```
typedef struct _CpaBufferList CpaBufferList
```

Scatter/Gather buffer list containing an array of flat buffers.

Description:

A scatter/gather buffer list structure. This buffer structure is typically used to represent a region of memory which is not physically contiguous, by describing it as a collection of buffers, each of which is physically contiguous.

Note

The memory for the pPrivateMetaData member must be allocated by the client as physically contiguous memory. When allocating memory for pPrivateMetaData, a call to the corresponding BufferListGetMetaSize function (e.g. cpaCyBufferListGetMetaSize) MUST be made to determine the size of the Meta Data Buffer. The returned size (in bytes) may then be passed in a memory allocation routine to allocate the pPrivateMetaData memory.

4.2.3.6 CpaPhysFlatBuffer

```
typedef struct _CpaPhysFlatBuffer CpaPhysFlatBuffer
```

Flat buffer structure with physical address.

Description:

Functions taking this structure do not need to do any virtual to physical address translation before writing the buffer to hardware.

4.2.3.7 CpaPhysBufferList

```
typedef struct _CpaPhysBufferList CpaPhysBufferList
```

Scatter/gather list containing an array of flat buffers with physical addresses.

Description:

Similar to CpaBufferList, this buffer structure is typically used to represent a region of memory which is not physically contiguous, by describing it as a collection of buffers, each of which is physically contiguous. The difference is that, in this case, the individual "flat" buffers are represented using physical, rather than virtual, addresses.

4.2.3.8 CpaStatus

```
typedef Cpa32S CpaStatus
```

API status value type definition

Description:

This type definition is used for the return values used in all the API functions. Common values are defined, for example see CPA_STATUS_SUCCESS, CPA_STATUS_FAIL, etc.

Definition at line 256 of file cpa.h.

4.2.3.9 CPA_DEPRECATED

```
typedef struct _CpaInstanceInfo CPA_DEPRECATED
```

Instance Types

Deprecated As of v1.3 of the Crypto API, this enum has been deprecated, replaced by CpaAccelerationServiceType.

Description:

Enumeration of the different instance types.

Instance State

Deprecated As of v1.3 of the Crypto API, this enum has been deprecated, replaced by CpaOperationalState.

Description:

Enumeration of the different instance states that are possible.

Instance Info Structure

Deprecated As of v1.3 of the Crypto API, this structure has been deprecated, replaced by CpaInstanceInfo2.

Description:

Structure that contains the information to describe the instance.

4.2.3.10 CpaAccelerationServiceType

```
typedef enum _CpaAccelerationServiceType CpaAccelerationServiceType
```

Service Type

Description:

Enumeration of the different service types.

4.2.3.11 CpaOperationalState

```
typedef enum _CpaOperationalState CpaOperationalState
```

Instance operational state

Description:

Enumeration of the different operational states that are possible.

4.2.3.12 CpaPhysicalInstanceId

```
typedef struct _CpaPhysicalInstanceId CpaPhysicalInstanceId
```

Physical Instance ID

Description:

Identifies the physical instance of an accelerator execution engine.

Accelerators grouped into "packages". Each accelerator can in turn contain one or more execution engines. Implementations of this API will define the packageId, acceleratorId, executionEngineId and busAddress as appropriate for the implementation. For example, for hardware-based accelerators, the packageId might identify the chip, which might contain multiple accelerators, each of which might contain multiple execution engines. The combination of packageId, acceleratorId and executionEngineId uniquely identifies the instance.

Hardware based accelerators implementing this API may also provide information on the location of the accelerator in the busAddress field. This field will be defined as appropriate for the implementation. For example, for PCIe attached accelerators, the busAddress may contain the PCIe bus, device and function number of the accelerators.

4.2.3.13 CpaInstanceInfo2

```
typedef struct _CpaInstanceInfo2 CpaInstanceInfo2
```

Instance Info Structure, version 2

Description:

Structure that contains the information to describe the instance.

4.2.3.14 CpaInstanceEvent

```
typedef enum _CpaInstanceEvent CpaInstanceEvent
```

Instance Events

Description:

Enumeration of the different events that will cause the registered Instance notification callback function to be invoked.

4.2.4 Enumeration Type Documentation

4.2.4.1 _CpaInstanceType

```
enum _CpaInstanceType
```

Instance Types

Deprecated As of v1.3 of the Crypto API, this enum has been deprecated, replaced by CpaAccelerationServiceType.

Description:

Enumeration of the different instance types.

Enumerator

CPA_INSTANCE_TYPE_CRYPTO	Cryptographic instance type
CPA_INSTANCE_TYPE_DATA_COMPRESSION	Data compression instance type
CPA_INSTANCE_TYPE_RAID	RAID instance type
CPA_INSTANCE_TYPE_XML	XML instance type
CPA_INSTANCE_TYPE_REGEX	Regular Expression instance type

Definition at line 357 of file cpa.h.

4.2.4.2 _CpaAccelerationServiceType

```
enum _CpaAccelerationServiceType
```

Service Type

Description:

Enumeration of the different service types.

Enumerator

CPA_ACC_SVC_TYPE_CRYPTO	Cryptography
CPA_ACC_SVC_TYPE_DATA_COMPRESSION	Data Compression
CPA_ACC_SVC_TYPE_PATTERN_MATCH	Pattern Match
CPA_ACC_SVC_TYPE_RAID	RAID
CPA_ACC_SVC_TYPE_XML	XML
CPA_ACC_SVC_TYPE_VIDEO_ANALYTICS	Video Analytics
CPA_ACC_SVC_TYPE_CRYPTO_ASYM	Cryptography - Asymmetric service
CPA_ACC_SVC_TYPE_CRYPTO_SYM	Cryptography - Symmetric service

Definition at line 379 of file cpa.h.

4.2.4.3 _CpaInstanceState

`enum _CpaInstanceState`

Instance State

Deprecated As of v1.3 of the Crypto API, this enum has been deprecated, replaced by CpaOperationalState.

Description:

Enumeration of the different instance states that are possible.

Enumerator

CPA_INSTANCE_STATE_INITIALISED	Instance is in the initialized state and ready for use.
CPA_INSTANCE_STATE_SHUTDOWN	Instance is in the shutdown state and not available for use.

Definition at line 412 of file cpa.h.

4.2.4.4 _CpaOperationalState

`enum _CpaOperationalState`

Instance operational state

Description:

Enumeration of the different operational states that are possible.

Enumerator

CPA_OPER_STATE_DOWN	Instance is not available for use. May not yet be initialized, or stopped.
CPA_OPER_STATE_UP	Instance is available for use. Has been initialized and started.

Definition at line 428 of file cpa.h.

4.2.4.5 _CpaInstanceEvent

```
enum _CpaInstanceEvent
```

Instance Events

Description:

Enumeration of the different events that will cause the registered Instance notification callback function to be invoked.

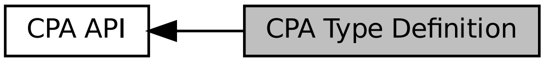
Enumerator

CPA_INSTANCE_EVENT_RESTARTING	Event type that triggers the registered instance notification callback function when and instance is restarting. The reason why an instance is restarting is implementation specific. For example a hardware implementation may send this event if the hardware device is about to be reset.
CPA_INSTANCE_EVENT_RESTARTED	Event type that triggers the registered instance notification callback function when and instance has restarted. The reason why an instance has restarted is implementation specific. For example a hardware implementation may send this event after the hardware device has been reset.
CPA_INSTANCE_EVENT_FATAL_ERROR	Event type that triggers the registered instance notification callback function when an error has been detected that requires the device to be reset. This event will be sent by all instances using the device, both on the host and guests.

Definition at line 622 of file cpa.h.

4.3 CPA Type Definition

Collaboration diagram for CPA Type Definition:



Macros

- #define NULL
- #define CPA_BITMAP(name, sizeInBits)
- #define CPA_BITMAP_BIT_TEST(bitmask, bit)
- #define CPA_BITMAP_BIT_SET(bitmask, bit)
- #define CPA_BITMAP_BIT_CLEAR(bitmask, bit)
- #define CPA_DEPRECATED

Typedefs

- typedef uint8_t Cpa8U
- typedef int8_t Cpa8S
- typedef uint16_t Cpa16U
- typedef int16_t Cpa16S
- typedef uint32_t Cpa32U
- typedef int32_t Cpa32S
- typedef uint64_t Cpa64U
- typedef int64_t Cpa64S
- typedef enum _CpaBoolean CpaBoolean

Enumerations

- enum _CpaBoolean

4.3.1 Detailed Description

File: cpa_types.h

Description:

This is the CPA Type Definitions.

4.3.2 Macro Definition Documentation

4.3.2.1 NULL

```
#define NULL
```

File: cpa_types.h

NULL definition.

Definition at line 119 of file cpa_types.h.

4.3.2.2 CPA_BITMAP

```
#define CPA_BITMAP(
              name,
              sizeInBits )
```

Declare a bitmap of specified size (in bits).

Description:

This macro is used to declare a bitmap of arbitrary size.

To test whether a bit in the bitmap is set, use CPA_BITMAP_BIT_TEST.

While most uses of bitmaps on the API are read-only, macros are also provided to set (see CPA_BITMAP_BIT_SET) and clear (see CPA_BITMAP_BIT_CLEAR) bits in the bitmap.

Definition at line 158 of file cpa_types.h.

4.3.2.3 CPA_BITMAP_BIT_TEST

```
#define CPA_BITMAP_BIT_TEST(
              bitmask,
              bit )
```

Test a specified bit in the specified bitmap. The bitmap may have been declared using CPA_BITMAP. Returns a Boolean (true if the bit is set, false otherwise).

Definition at line 161 of file cpa_types.h.

4.3.2.4 CPA_BITMAP_BIT_SET

```
#define CPA_BITMAP_BIT_SET(
              bitmask,
              bit )
```

File: cpa_types.h

Set a specified bit in the specified bitmap. The bitmap may have been declared using CPA_BITMAP.

Definition at line 171 of file cpa_types.h.

4.3.2.5 CPA_BITMAP_BIT_CLEAR

```
#define CPA_BITMAP_BIT_CLEAR(
              bitmask,
              bit )
```

Clear a specified bit in the specified bitmap. The bitmap may have been declared using CPA_BITMAP.

Definition at line 181 of file cpa_types.h.

4.3.2.6 CPA_DEPRECATED

```
typedef struct _CpaCyEcPointVerifyOpData CPA_DEPRECATED
```

Description:

Declare a function or type and mark it as deprecated so that usages get flagged with a warning.

Instance State

Deprecated As of v1.3 of the Crypto API, this enum has been deprecated, replaced by CpaOperationalState.

Description:

Enumeration of the different instance states that are possible.

Instance Info Structure

Deprecated As of v1.3 of the Crypto API, this structure has been deprecated, replaced by CpaInstanceInfo2.

Description:

Structure that contains the information to describe the instance.

EC Point Verification Operation Data.

Description:

This structure contains the operation data for the cpaCyEcPointVerify function. The client MUST allocate the memory for this structure and the items pointed to by this structure. When the structure is passed into the function, ownership of the memory passes to the function. Ownership of the memory returns to the client when this structure is returned in the callback function.

For optimal performance all data buffers SHOULD be 8-byte aligned.

All values in this structure are required to be in Most Significant Byte first order, e.g. a.pData[0] = MSB.

Note

If the client modifies or frees the memory referenced in this structure after it has been submitted to the CpaCyEcPointVerify function, and before it has been returned in the callback, undefined behavior will result.

See also

cpaCyEcPointVerify()

Definition at line 219 of file cpa_types.h.

4.3.3 Typedef Documentation

4.3.3.1 Cpa8U

```
typedef uint8_t Cpa8U
```

File: cpa_types.h

Unsigned byte base type.

Definition at line 74 of file cpa_types.h.

4.3.3.2 Cpa8S

```
typedef int8_t Cpa8S
```

File: cpa_types.h

Signed byte base type.

Definition at line 79 of file cpa_types.h.

4.3.3.3 Cpa16U

```
typedef uint16_t Cpa16U
```

File: cpa_types.h

Unsigned double-byte base type.

Definition at line 84 of file cpa_types.h.

4.3.3.4 Cpa16S

```
typedef int16_t Cpa16S
```

File: cpa_types.h

Signed double-byte base type.

Definition at line 89 of file cpa_types.h.

4.3.3.5 Cpa32U

```
typedef uint32_t Cpa32U
```

File: cpa_types.h

Unsigned quad-byte base type.

Definition at line 94 of file cpa_types.h.

4.3.3.6 Cpa32S

```
typedef int32_t Cpa32S
```

File: cpa_types.h

Signed quad-byte base type.

Definition at line 99 of file cpa_types.h.

4.3.3.7 Cpa64U

```
typedef uint64_t Cpa64U
```

File: cpa_types.h

Unsigned double-quad-byte base type.

Definition at line 104 of file cpa_types.h.

4.3.3.8 Cpa64S

```
typedef int64_t Cpa64S
```

File: cpa_types.h

Signed double-quad-byte base type.

Definition at line 109 of file cpa_types.h.

4.3.3.9 CpaBoolean

```
typedef enum _CpaBoolean CpaBoolean
```

Boolean type.

Description:

Functions in this API use this type for Boolean variables that take true or false values.

4.3.4 Enumeration Type Documentation

4.3.4.1 _CpaBoolean

`enum _CpaBoolean`

Boolean type.

Description:

Functions in this API use this type for Boolean variables that take true or false values.

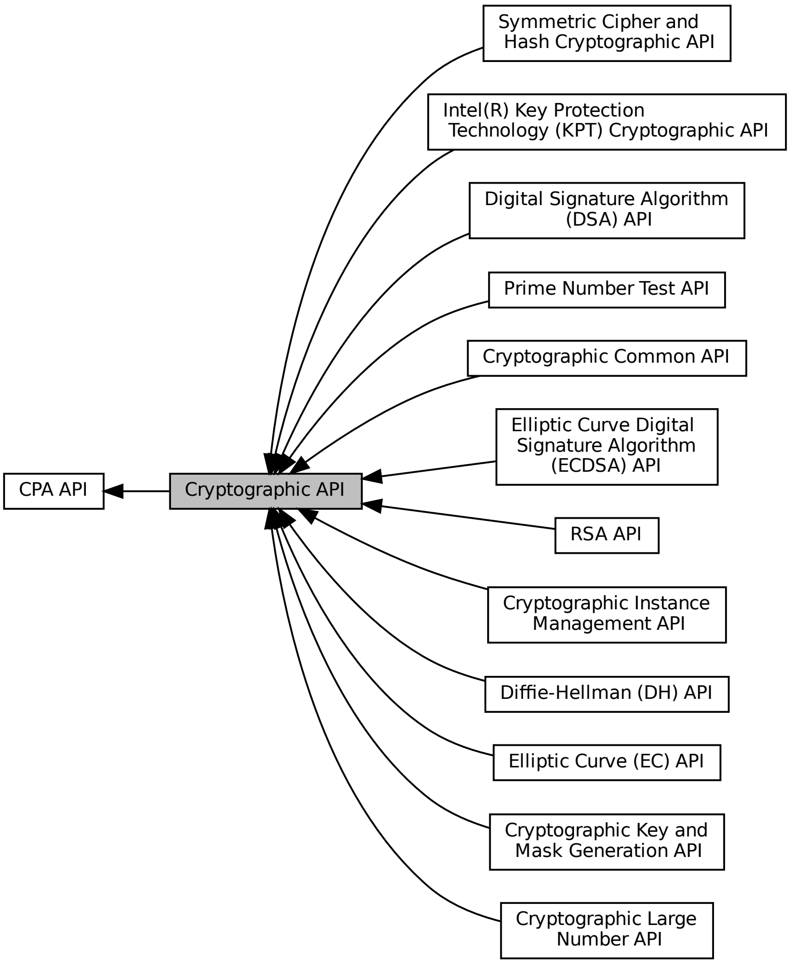
Enumerator

CPA_FALSE	False value
CPA_TRUE	True value

Definition at line 136 of file cpa_types.h.

4.4 Cryptographic API

Collaboration diagram for Cryptographic API:



Modules

- Cryptographic Common API
- Cryptographic Instance Management API
- Symmetric Cipher and Hash Cryptographic API
- Cryptographic Key and Mask Generation API
- RSA API
- Diffie-Hellman (DH) API
- Digital Signature Algorithm (DSA) API
- Elliptic Curve (EC) API
- Elliptic Curve Digital Signature Algorithm (ECDSA) API
- Cryptographic Large Number API
- Prime Number Test API
- Intel(R) Key Protection Technology (KPT) Cryptographic API

4.4.1 Detailed Description

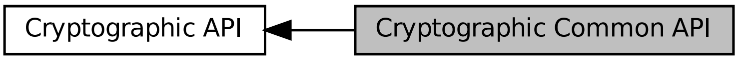
File: cpa_cy_common.h

Description:

These functions specify the Cryptographic API.

4.5 Cryptographic Common API

Collaboration diagram for Cryptographic Common API:



Typedefs

- typedef enum _CpaCyPriority CpaCyPriority
- typedef void(∗ CpaCyGenericCbFunc) (void ∗pCallbackTag, CpaStatus status, void ∗pOpData)
- typedef void(∗ CpaCyGenFlatBufCbFunc) (void ∗pCallbackTag, CpaStatus status, void ∗pOpdata, CpaFlatBuffer ∗pOut)
- typedef void(∗ CpaCyInstanceNotificationCbFunc) (const CpaInstanceHandle instanceHandle, void ∗pCallbackTag, const CpaInstanceEvent instanceEvent)

Enumerations

- enum _CpaCyPriority

Functions

- CpaStatus cpaCyBufferListGetMetaSize (const CpaInstanceHandle instanceHandle, Cpa32U numBuffers, Cpa32U ∗pSizeInBytes)
- CpaStatus cpaCyGetStatusText (const CpaInstanceHandle instanceHandle, CpaStatus errStatus, Cpa8S ∗pStatusText)
- CpaStatus cpaCyGetNumInstances (Cpa16U ∗pNumInstances)
- CpaStatus cpaCyGetInstances (Cpa16U numInstances, CpaInstanceHandle ∗cyInstances)
- CpaStatus CPA_DEPRECATED cpaCyInstanceGetInfo (const CpaInstanceHandle instanceHandle, struct _CpaInstanceInfo ∗pInstanceInfo)
- CpaStatus cpaCyInstanceGetInfo2 (const CpaInstanceHandle instanceHandle, CpaInstanceInfo2 ∗pInstanceInfo2)
- CpaStatus cpaCyInstanceSetNotificationCb (const CpaInstanceHandle instanceHandle, const CpaCyInstanceNotificationCbFunc pInstanceNotificationCb, void ∗pCallbackTag)

4.5.1 Detailed Description

File: cpa_cy_common.h

Description:

This file specifies items which are common for both the asymmetric (public key cryptography) and the symmetric operations for the Cryptographic API.

4.5.2 Typedef Documentation

4.5.2.1 CpaCyPriority

```
typedef enum _CpaCyPriority CpaCyPriority
```

File: cpa_cy_common.h

Request priority

Description:

Enumeration of priority of the request to be given to the API. Currently two levels - HIGH and NORMAL are supported. HIGH priority requests will be prioritized on a "best-effort" basis over requests that are marked with a NORMAL priority.

4.5.2.2 CpaCyGenericCbFunc

```
typedef void(* CpaCyGenericCbFunc) (void *pCallbackTag, CpaStatus status, void *pOpData)
```

Definition of the crypto generic callback function

Description:

This data structure specifies the prototype for a generic callback function

Context:

This callback function can be executed in a context that DOES NOT permit sleeping to occur.

Assumptions:

None

Side-Effects:

None

Reentrant:

No

Thread-safe:

Yes

Parameters

in	*pCallbackTag*	Opaque value provided by user while making individual function call.
in	*status*	Status of the operation. Valid values are CPA_STATUS_SUCCESS, CPA_STATUS_FAIL and CPA_STATUS_UNSUPPORTED.
in	*pOpData*	Opaque Pointer to the operation data that was submitted in the request

Return values

None	

Precondition

Component has been initialized.

Postcondition

None

Note

None

See also

cpaCyKeyGenSsl()

Definition at line 168 of file cpa_cy_common.h.

4.5.2.3 CpaCyGenFlatBufCbFunc

```
typedef void(* CpaCyGenFlatBufCbFunc) (void *pCallbackTag, CpaStatus status, void *pOpdata,
CpaFlatBuffer *pOut)
```

Definition of generic callback function with an additional output CpaFlatBuffer parameter.

Description:

This data structure specifies the prototype for a generic callback function which provides an output buffer (of type CpaFlatBuffer).

Context:

This callback function can be executed in a context that DOES NOT permit sleeping to occur.

Assumptions:

None

Side-Effects:

None

Reentrant:

No

Thread-safe:

Yes

Parameters

in	*pCallbackTag*	Opaque value provided by user while making individual function call.
in	*status*	Status of the operation. Valid values are CPA_STATUS_SUCCESS, CPA_STATUS_FAIL and CPA_STATUS_UNSUPPORTED.
in	*pOpData*	Opaque Pointer to the operation data that was submitted in the request
in	*pOut*	Pointer to the output buffer provided in the request invoking this callback.

Return values

None	

Precondition

Component has been initialized.

Postcondition

None

Note

None

See also

None

Definition at line 217 of file cpa_cy_common.h.

4.5.2.4 CpaCyInstanceNotificationCbFunc

```
typedef void(* CpaCyInstanceNotificationCbFunc) (const CpaInstanceHandle instanceHandle, void
*pCallbackTag, const CpaInstanceEvent instanceEvent)
```

Callback function for instance notification support.

Description:

This is the prototype for the instance notification callback function. The callback function is passed in as a parameter to the cpaCyInstanceSetNotificationCb function.

Context:

This function will be executed in a context that requires that sleeping MUST NOT be permitted.

Assumptions:

None

Side-Effects:

None

Blocking:

No

Reentrant:

No

Thread-safe:

Yes

Parameters

in	*instanceHandle*	Instance handle.
in	*pCallbackTag*	Opaque value provided by user while making individual function calls.
in	*instanceEvent*	The event that will trigger this function to get invoked.

Return values

None	

Precondition

Component has been initialized and the notification function has been set via the cpaCyInstanceSetNotificationCb function.

Postcondition

None

Note

None

See also

cpaCyInstanceSetNotificationCb(),

Definition at line 594 of file cpa_cy_common.h.

4.5.3 Enumeration Type Documentation

4.5.3.1 _CpaCyPriority

```
enum _CpaCyPriority
```

File: cpa_cy_common.h

Request priority

Description:

Enumeration of priority of the request to be given to the API. Currently two levels - HIGH and NORMAL are supported. HIGH priority requests will be prioritized on a "best-effort" basis over requests that are marked with a NORMAL priority.

Enumerator

CPA_CY_PRIORITY_NORMAL	Normal priority
CPA_CY_PRIORITY_HIGH	High priority

Definition at line 117 of file cpa_cy_common.h.

4.5.4 Function Documentation

4.5.4.1 cpaCyBufferListGetMetaSize()

```
CpaStatus cpaCyBufferListGetMetaSize (
            const CpaInstanceHandle instanceHandle,
            Cpa32U numBuffers,
            Cpa32U * pSizeInBytes )
```

Function to return the size of the memory which must be allocated for the pPrivateMetaData member of CpaBufferList.

Description:

This function is used obtain the size (in bytes) required to allocate a buffer descriptor for the pPrivateMetaData member in the CpaBufferList the structure. Should the function return zero then no meta data is required for the buffer list.

Context:

This function may be called from any context.

Assumptions:

None

Side-Effects:

None

Blocking:

No

Reentrant:

No

Thread-safe:

Yes

Parameters

in	*instanceHandle*	Handle to an instance of this API.
in	*numBuffers*	The number of pointers in the CpaBufferList. this is the maximum number of CpaFlatBuffers which may be contained in this CpaBufferList.
out	*pSizeInBytes*	Pointer to the size in bytes of memory to be allocated when the client wishes to allocate a cpaFlatBuffer

Return values

CPA_STATUS_SUCCESS	Function executed successfully.
CPA_STATUS_FAIL	Function failed.
CPA_STATUS_INVALID_PARAM	Invalid parameter passed in.
CPA_STATUS_UNSUPPORTED	Function is not supported.

Precondition

None.

Postcondition

None

Note

None

See also

cpaCyGetInstances()

4.5.4.2 cpaCyGetStatusText()

```
CpaStatus cpaCyGetStatusText (
            const CpaInstanceHandle instanceHandle,
            CpaStatus errStatus,
            Cpa8S * pStatusText )
```

Function to return a string indicating the specific error that occurred for a particular instance.

Description:

When a function invocation on a particular instance returns an error, the client can invoke this function to query the instance for a null terminated string which describes the general error condition, and if available additional text on the specific error. The Client MUST allocate CPA_STATUS_MAX_STR_LENGTH_IN_BYTES bytes for the buffer string.

Context:

This function may be called from any context.

Assumptions:

None

Side-Effects:

None

Blocking:

No

Reentrant:

No

Thread-safe:

Yes

Parameters

in	*instanceHandle*	Handle to an instance of this API.
in	*errStatus*	The error condition that occurred
out	*pStatusText*	Pointer to the string buffer that will be updated with a null terminated status text string. The invoking application MUST allocate this buffer to be CPA_STATUS_MAX_STR_LENGTH_IN_BYTES.

Return values

CPA_STATUS_SUCCESS	Function executed successfully.
CPA_STATUS_FAIL	Function failed. Note, In this scenario it is INVALID to call this function a further time.
CPA_STATUS_INVALID_PARAM	Invalid parameter passed in.
CPA_STATUS_UNSUPPORTED	Function is not supported.

Precondition

None.

Postcondition

None

Note

None

See also

CpaStatus

4.5.4.3 cpaCyGetNumInstances()

```
CpaStatus cpaCyGetNumInstances (
            Cpa16U * pNumInstances )
```

Get the number of instances that are supported by the API implementation.

Description:

This function will get the number of instances that are supported by an implementation of the Cryptographic API. This number is then used to determine the size of the array that must be passed to cpaCyGetInstances().

Context:

This function MUST NOT be called from an interrupt context as it MAY sleep.

Assumptions:

None

Side-Effects:

None

Blocking:

This function is synchronous and blocking.

Reentrant:

No

Thread-safe:

Yes

Parameters

out	*pNumInstances*	Pointer to where the number of instances will be written.

Return values

CPA_STATUS_SUCCESS	Function executed successfully.
CPA_STATUS_FAIL	Function failed.
CPA_STATUS_INVALID_PARAM	Invalid parameter passed in.
CPA_STATUS_UNSUPPORTED	Function is not supported.

Precondition

None

Postcondition

None

Note

This function operates in a synchronous manner and no asynchronous callback will be generated

See also

cpaCyGetInstances

4.5.4.4 cpaCyGetInstances()

```
CpaStatus cpaCyGetInstances (
            Cpa16U numInstances,
            CpaInstanceHandle * cyInstances )
```

Get the handles to the instances that are supported by the API implementation.

Description:

This function will return handles to the instances that are supported by an implementation of the Cryptographic API. These instance handles can then be used as input parameters with other Cryptographic API functions.

This function will populate an array that has been allocated by the caller. The size of this API will have been determined by the cpaCyGetNumInstances() function.

Context:

This function MUST NOT be called from an interrupt context as it MAY sleep.

Assumptions:

None

Side-Effects:

None

Blocking:

This function is synchronous and blocking.

Reentrant:

No

Thread-safe:

Yes

Parameters

in	*numInstances*	Size of the array. If the value is not the same as the number of instances supported, then an error (CPA_STATUS_INVALID_PARAM) is returned.
in,out	*cyInstances*	Pointer to where the instance handles will be written.

Return values

CPA_STATUS_SUCCESS	Function executed successfully.

Return values

CPA_STATUS_FAIL	Function failed.
CPA_STATUS_INVALID_PARAM	Invalid parameter passed in.
CPA_STATUS_UNSUPPORTED	Function is not supported.

Precondition

None

Postcondition

None

Note

This function operates in a synchronous manner and no asynchronous callback will be generated

See also

cpaCyGetNumInstances

4.5.4.5 cpaCyInstanceGetInfo()

```
CpaStatus CPA_DEPRECATED cpaCyInstanceGetInfo (
            const CpaInstanceHandle instanceHandle,
            struct _CpaInstanceInfo * pInstanceInfo )
```

Function to get information on a particular instance.

Deprecated As of v1.3 of the Crypto API, this function has been deprecated, replaced by cpaCyInstanceGetInfo2.

Description:

This function will provide instance specific information through a CpaInstanceInfo structure.

Context:

This function may be called from any context.

Assumptions:

None

Side-Effects:

None

Blocking:

No

Reentrant:

No

Thread-safe:

Yes

Parameters

`in`	*instanceHandle*	Handle to an instance of this API to be initialized.
`out`	*pInstanceInfo*	Pointer to the memory location allocated by the client into which the CpaInstanceInfo structure will be written.

Return values

CPA_STATUS_SUCCESS	Function executed successfully.
CPA_STATUS_FAIL	Function failed.
CPA_STATUS_INVALID_PARAM	Invalid parameter passed in.
CPA_STATUS_UNSUPPORTED	Function is not supported.

Precondition

The client has retrieved an instanceHandle from successive calls to cpaCyGetNumInstances and cpaCyGetInstances.

Postcondition

None

Note

None

See also

cpaCyGetNumInstances, cpaCyGetInstances, CpaInstanceInfo

4.5.4.6 cpaCyInstanceGetInfo2()

```
CpaStatus cpaCyInstanceGetInfo2 (
            const CpaInstanceHandle instanceHandle,
            CpaInstanceInfo2 * pInstanceInfo2 )
```

Function to get information on a particular instance.

Description:

This function will provide instance specific information through a CpaInstanceInfo2 structure. Supersedes cpaCyInstanceGetInfo.

Context:

This function may be called from any context.

Assumptions:

None

Side-Effects:

None

Blocking:

No

Reentrant:

No

Thread-safe:

Yes

Parameters

in	*instanceHandle*	Handle to an instance of this API to be initialized.
out	*pInstanceInfo2*	Pointer to the memory location allocated by the client into which the CpaInstanceInfo2 structure will be written.

Return values

CPA_STATUS_SUCCESS	Function executed successfully.
CPA_STATUS_FAIL	Function failed.
CPA_STATUS_INVALID_PARAM	Invalid parameter passed in.
CPA_STATUS_UNSUPPORTED	Function is not supported.

Precondition

The client has retrieved an instanceHandle from successive calls to cpaCyGetNumInstances and cpaCyGetInstances.

Postcondition

None

Note

None

See also

cpaCyGetNumInstances, cpaCyGetInstances, CpaInstanceInfo

4.5.4.7 cpaCyInstanceSetNotificationCb()

```
CpaStatus cpaCyInstanceSetNotificationCb (
            const CpaInstanceHandle instanceHandle,
            const CpaCyInstanceNotificationCbFunc pInstanceNotificationCb,
            void * pCallbackTag )
```

Subscribe for instance notifications.

Description:

Clients of the CpaCy interface can subscribe for instance notifications by registering a CpaCyInstanceNotificationCbFunc function.

Context:

This function may be called from any context.

Assumptions:

None

Side-Effects:

None

Blocking:

No

Reentrant:

No

Thread-safe:

Yes

Parameters

in	*instanceHandle*	Instance handle.
in	*pInstanceNotificationCb*	Instance notification callback function pointer.
in	*pCallbackTag*	Opaque value provided by user while making individual function calls.

Return values

CPA_STATUS_SUCCESS	Function executed successfully.
CPA_STATUS_FAIL	Function failed.
CPA_STATUS_INVALID_PARAM	Invalid parameter passed in.
CPA_STATUS_UNSUPPORTED	Function is not supported.

Precondition

Instance has been initialized.

Postcondition

None

Note

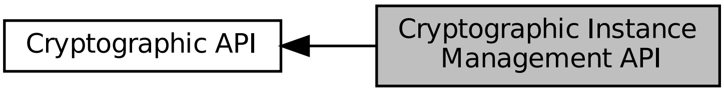
None

See also

CpaCyInstanceNotificationCbFunc

4.6 Cryptographic Instance Management API

Collaboration diagram for Cryptographic Instance Management API:



Data Structures

- struct _CpaCyCapabilitiesInfo

Typedefs

- typedef struct _CpaCyCapabilitiesInfo CpaCyCapabilitiesInfo

Functions

- CpaStatus cpaCyStartInstance (CpaInstanceHandle instanceHandle)
- CpaStatus cpaCyStopInstance (CpaInstanceHandle instanceHandle)
- CpaStatus cpaCyQueryCapabilities (const CpaInstanceHandle instanceHandle, CpaCyCapabilitiesInfo ∗pCapInfo)
- CpaStatus cpaCySetAddressTranslation (const CpaInstanceHandle instanceHandle, CpaVirtualToPhysical virtual2Physical)

4.6.1 Detailed Description

File: cpa_cy_im.h

Description:

These functions specify the Instance Management API for available Cryptographic Instances. It is expected that these functions will only be called via a single system maintenance entity, rather than individual clients.

4.6.2 Typedef Documentation

4.6.2.1 CpaCyCapabilitiesInfo

```
typedef struct _CpaCyCapabilitiesInfo CpaCyCapabilitiesInfo
```

Cryptographic Capabilities Info

Description:

This structure contains the capabilities that vary across API implementations. This structure is used in conjunction with cpaCyQueryCapabilities() to determine the capabilities supported by a particular API implementation.

The client MUST allocate memory for this structure and any members that require memory. When the structure is passed into the function ownership of the memory passes to the function. Ownership of the memory returns to the client when the function returns.

4.6.3 Function Documentation

4.6.3.1 cpaCyStartInstance()

```
CpaStatus cpaCyStartInstance (
            CpaInstanceHandle instanceHandle )
```

Cryptographic Component Initialization and Start function.

Description:

This function will initialize and start the Cryptographic component. It MUST be called before any other crypto function is called. This function SHOULD be called only once (either for the very first time, or after an cpaCyStopInstance call which succeeded) per instance. Subsequent calls will have no effect.

Context:

This function may sleep, and MUST NOT be called in interrupt context.

Assumptions:

None

Side-Effects:

None

Blocking:

This function is synchronous and blocking.

Reentrant:

No

Thread-safe:

Yes

Parameters

out	*instanceHandle*	Handle to an instance of this API to be initialized.

Return values

CPA_STATUS_SUCCESS	Function executed successfully.
CPA_STATUS_FAIL	Function failed. Suggested course of action is to shutdown and restart.
CPA_STATUS_UNSUPPORTED	Function is not supported.

Precondition

None.

Postcondition

None

Note

Note that this is a synchronous function and has no completion callback associated with it.

See also

cpaCyStopInstance()

4.6.3.2 cpaCyStopInstance()

```
CpaStatus cpaCyStopInstance (
            CpaInstanceHandle instanceHandle )
```

Cryptographic Component Stop function.

Description:

This function will stop the Cryptographic component and free all system resources associated with it. The client MUST ensure that all outstanding operations have completed before calling this function. The recommended approach to ensure this is to deregister all session or callback handles before calling this function. If outstanding operations still exist when this function is invoked, the callback function for each of those operations will NOT be invoked and the shutdown will continue. If the component is to be restarted, then a call to cpaCyStartInstance is required.

Context:

This function may sleep, and so MUST NOT be called in interrupt context.

Assumptions:

None

Side-Effects:

None

Blocking:

This function is synchronous and blocking.

Reentrant:

No

Thread-safe:

Yes

Parameters

in	*instanceHandle*	Handle to an instance of this API to be shutdown.

Return values

CPA_STATUS_SUCCESS	Function executed successfully.
CPA_STATUS_FAIL	Function failed. Suggested course of action is to ensure requests are not still being submitted and that all sessions are deregistered. If this does not help, then forcefully remove the component from the system.
CPA_STATUS_UNSUPPORTED	Function is not supported.

Precondition

The component has been initialized via cpaCyStartInstance.

Postcondition

None

Note

Note that this is a synchronous function and has no completion callback associated with it.

See also

cpaCyStartInstance()

4.6.3.3 cpaCyQueryCapabilities()

```
CpaStatus cpaCyQueryCapabilities (
            const CpaInstanceHandle instanceHandle,
            CpaCyCapabilitiesInfo * pCapInfo )
```

Returns capabilities of a Cryptographic API instance

Description:

This function is used to query the instance capabilities.

Context:

The function shall not be called in an interrupt context.

Assumptions:

None

Side-Effects:

None

Blocking:

This function is synchronous and blocking.

Reentrant:

No

Thread-safe:

Yes

Parameters

in	*instanceHandle*	Handle to an instance of this API.
out	*pCapInfo*	Pointer to capabilities info structure. All fields in the structure are populated by the API instance.

Return values

CPA_STATUS_SUCCESS	Function executed successfully.
CPA_STATUS_FAIL	Function failed.
CPA_STATUS_INVALID_PARAM	Invalid parameter passed in.
CPA_STATUS_UNSUPPORTED	Function is not supported.

Precondition

The instance has been initialized via the cpaCyStartInstance function.

Postcondition

None

4.6.3.4 cpaCySetAddressTranslation()

```
CpaStatus cpaCySetAddressTranslation (
            const CpaInstanceHandle instanceHandle,
            CpaVirtualToPhysical virtual2Physical )
```

Sets the address translation function

Description:

This function is used to set the virtual to physical address translation routine for the instance. The specified routine is used by the instance to perform any required translation of a virtual address to a physical address. If the application does not invoke this function, then the instance will use its default method, such as virt2phys, for address translation.

Context:

The function shall not be called in an interrupt context.

Assumptions:

None

Side-Effects:

None

Blocking:

This function is synchronous and blocking.

Reentrant:

No

Thread-safe:

Yes

Parameters

in	*instanceHandle*	Handle to an instance of this API.
in	*virtual2Physical*	Routine that performs virtual to physical address translation.

Return values

CPA_STATUS_SUCCESS	Function executed successfully.
CPA_STATUS_FAIL	Function failed.
CPA_STATUS_INVALID_PARAM	Invalid parameter passed in.
CPA_STATUS_UNSUPPORTED	Function is not supported.

Precondition

None

Postcondition

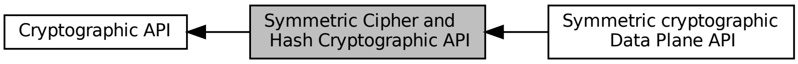
None

See also

None

4.7 Symmetric Cipher and Hash Cryptographic API

Collaboration diagram for Symmetric Cipher and Hash Cryptographic API:



Modules

- Symmetric cryptographic Data Plane API

Data Structures

- struct _CpaCySymCipherSetupData
- struct _CpaCySymHashNestedModeSetupData
- struct _CpaCySymHashAuthModeSetupData
- struct _CpaCySymHashSetupData
- struct _CpaCySymSessionSetupData
- struct _CpaCySymSessionUpdateData
- struct _CpaCySymOpData
- struct _CpaCySymStats
- struct _CpaCySymStats64
- struct _CpaCySymCapabilitiesInfo

Macros

- #define CPA_CY_SYM_CIPHER_CAP_BITMAP_SIZE
- #define CPA_CY_SYM_HASH_CAP_BITMAP_SIZE
- #define CPA_CY_SYM_CCM_SET_NONCE(pOpData, pNonce, nonceLen)
- #define CPA_CY_SYM_CCM_SET_AAD(pOpData, pAad, aadLen)

Typedefs

- typedef void ∗ CpaCySymSessionCtx
- typedef enum _CpaCySymPacketType CpaCySymPacketType
- typedef enum _CpaCySymOp CpaCySymOp
- typedef enum _CpaCySymCipherAlgorithm CpaCySymCipherAlgorithm
- typedef enum _CpaCySymCipherDirection CpaCySymCipherDirection
- typedef struct _CpaCySymCipherSetupData CpaCySymCipherSetupData
- typedef enum _CpaCySymHashMode CpaCySymHashMode
- typedef enum _CpaCySymHashAlgorithm CpaCySymHashAlgorithm
- typedef struct _CpaCySymHashNestedModeSetupData CpaCySymHashNestedModeSetupData
- typedef struct _CpaCySymHashAuthModeSetupData CpaCySymHashAuthModeSetupData
- typedef struct _CpaCySymHashSetupData CpaCySymHashSetupData
- typedef enum _CpaCySymAlgChainOrder CpaCySymAlgChainOrder
- typedef struct _CpaCySymSessionSetupData CpaCySymSessionSetupData
- typedef struct _CpaCySymSessionUpdateData CpaCySymSessionUpdateData
- typedef struct _CpaCySymOpData CpaCySymOpData
- typedef struct _CpaCySymStats CPA_DEPRECATED
- typedef struct _CpaCySymStats64 CpaCySymStats64
- typedef void(∗ CpaCySymCbFunc) (void ∗pCallbackTag, CpaStatus status, const CpaCySymOp operationType, void ∗pOpData, CpaBufferList ∗pDstBuffer, CpaBoolean verifyResult)
- typedef struct _CpaCySymCapabilitiesInfo CpaCySymCapabilitiesInfo

Enumerations

- enum _CpaCySymPacketType
- enum _CpaCySymOp
- enum _CpaCySymCipherAlgorithm
- enum _CpaCySymCipherDirection
- enum _CpaCySymHashMode
- enum _CpaCySymHashAlgorithm
- enum _CpaCySymAlgChainOrder

Functions

- CpaStatus cpaCySymSessionCtxGetSize (const CpaInstanceHandle instanceHandle, const CpaCySymSessionSetupData ∗pSessionSetupData, Cpa32U ∗pSessionCtxSizeInBytes)
- CpaStatus cpaCySymSessionCtxGetDynamicSize (const CpaInstanceHandle instanceHandle, const CpaCySymSessionSetupData ∗pSessionSetupData, Cpa32U ∗pSessionCtxSizeInBytes)
- CpaStatus cpaCySymInitSession (const CpaInstanceHandle instanceHandle, const CpaCySymCbFunc pSymCb, const CpaCySymSessionSetupData ∗pSessionSetupData, CpaCySymSessionCtx sessionCtx)
- CpaStatus cpaCySymRemoveSession (const CpaInstanceHandle instanceHandle, CpaCySymSessionCtx pSessionCtx)
- CpaStatus cpaCySymUpdateSession (CpaCySymSessionCtx sessionCtx, const CpaCySymSessionUpdateData ∗pSessionUpdateData)
- CpaStatus cpaCySymSessionInUse (CpaCySymSessionCtx sessionCtx, CpaBoolean ∗pSessionInUse)
- CpaStatus cpaCySymPerformOp (const CpaInstanceHandle instanceHandle, void ∗pCallbackTag, const CpaCySymOpData ∗pOpData, const CpaBufferList ∗pSrcBuffer, CpaBufferList ∗pDstBuffer, CpaBoolean ∗pVerifyResult)
- CpaStatus CPA_DEPRECATED cpaCySymQueryStats (const CpaInstanceHandle instanceHandle, struct _CpaCySymStats ∗pSymStats)
- CpaStatus cpaCySymQueryStats64 (const CpaInstanceHandle instanceHandle, CpaCySymStats64 ∗pSymStats)
- CpaStatus cpaCySymQueryCapabilities (const CpaInstanceHandle instanceHandle, CpaCySymCapabilitiesInfo ∗pCapInfo)

4.7.1 Detailed Description

File: cpa_cy_sym.h

Description:

These functions specify the Cryptographic API for symmetric cipher, hash, and combined cipher and hash operations.

4.7.2 Macro Definition Documentation

4.7.2.1 CPA_CY_SYM_CIPHER_CAP_BITMAP_SIZE

```
#define CPA_CY_SYM_CIPHER_CAP_BITMAP_SIZE
```

Size of bitmap needed for cipher "capabilities" type.

Description:

Defines the number of bits in the bitmap to represent supported ciphers in the type CpaCySymCapabilitiesInfo. Should be set to at least one greater than the largest value in the enumerated type CpaCySymHashAlgorithm, so that the value of the enum constant can also be used as the bit position in the bitmap.

A larger value was chosen to allow for extensibility without the need to change the size of the bitmap (to ease backwards compatibility in future versions of the API).

Definition at line 201 of file cpa_cy_sym.h.

4.7.2.2 CPA_CY_SYM_HASH_CAP_BITMAP_SIZE

```
#define CPA_CY_SYM_HASH_CAP_BITMAP_SIZE
```

Size of bitmap needed for hash "capabilities" type.

Description:

Defines the number of bits in the bitmap to represent supported hashes in the type CpaCySymCapabilitiesInfo. Should be set to at least one greater than the largest value in the enumerated type CpaCySymHashAlgorithm, so that the value of the enum constant can also be used as the bit position in the bitmap.

A larger value was chosen to allow for extensibility without the need to change the size of the bitmap (to ease backwards compatibility in future versions of the API).

Definition at line 402 of file cpa_cy_sym.h.

4.7.2.3 CPA_CY_SYM_CCM_SET_NONCE

```
#define CPA_CY_SYM_CCM_SET_NONCE(
              pOpData,
              pNonce,
              nonceLen )
```

Setup the nonce for CCM.

Description:

This macro sets the nonce in the appropriate locations of the CpaCySymOpData struct for the authenticated encryption algorithm CPA_CY_SYM_HASH_AES_CCM.

Definition at line 922 of file cpa_cy_sym.h.

4.7.2.4 CPA_CY_SYM_CCM_SET_AAD

```
#define CPA_CY_SYM_CCM_SET_AAD(
            pOpData,
            pAad,
            aadLen )
```

Setup the additional authentication data for CCM.

Description:

This macro sets the additional authentication data in the appropriate location of theCpaCySymOpData struct for the authenticated encryptionalgorithm CPA_CY_SYM_HASH_AES_CCM.

Definition at line 936 of file cpa_cy_sym.h.

4.7.3 Typedef Documentation

4.7.3.1 CpaCySymSessionCtx

```
typedef void* CpaCySymSessionCtx
```

Cryptographic component symmetric session context handle.

Description:

Handle to a cryptographic session context. The memory for this handle is allocated by the client. The size of the memory that the client needs to allocate is determined by a call to the cpaCySymSessionCtxGetSize or cpaCySymSessionCtxGetDynamicSize functions. The session context memory is initialized with a call to the cpaCySymInitSession function. This memory MUST not be freed until a call to cpaCySymRemoveSession has completed successfully.

Definition at line 50 of file cpa_cy_sym.h.

4.7.3.2 CpaCySymPacketType

```
typedef enum _CpaCySymPacketType CpaCySymPacketType
```

Packet type for the cpaCySymPerformOp function

Description:

Enumeration which is used to indicate to the symmetric cryptographic perform function on which type of packet the operation is required to be invoked. Multi-part cipher and hash operations are useful when processing needs to be performed on a message which is available to the client in multiple parts (for example due to network fragmentation of the packet).

Note

There are some restrictions regarding the operations on which partial packet processing is supported. For details, see the function cpaCySymPerformOp.

See also

cpaCySymPerformOp()

4.7.3.3 CpaCySymOp

```
typedef enum _CpaCySymOp CpaCySymOp
```

Types of operations supported by the cpaCySymPerformOp function.

Description:

This enumeration lists different types of operations supported by the cpaCySymPerformOp function. The operation type is defined during session registration and cannot be changed for a session once it has been setup.

See also

cpaCySymPerformOp

4.7.3.4 CpaCySymCipherAlgorithm

```
typedef enum _CpaCySymCipherAlgorithm CpaCySymCipherAlgorithm
```

Cipher algorithms.

Description:

This enumeration lists supported cipher algorithms and modes.

4.7.3.5 CpaCySymCipherDirection

```
typedef enum _CpaCySymCipherDirection CpaCySymCipherDirection
```

Symmetric Cipher Direction

Description:

This enum indicates the cipher direction (encryption or decryption).

4.7.3.6 CpaCySymCipherSetupData

```
typedef struct _CpaCySymCipherSetupData CpaCySymCipherSetupData
```

Symmetric Cipher Setup Data.

Description:

This structure contains data relating to Cipher (Encryption and Decryption) to set up a session.

4.7.3.7 CpaCySymHashMode

```
typedef enum _CpaCySymHashMode CpaCySymHashMode
```

Symmetric Hash mode

Description:

This enum indicates the Hash Mode.

4.7.3.8 CpaCySymHashAlgorithm

```
typedef enum _CpaCySymHashAlgorithm CpaCySymHashAlgorithm
```

Hash algorithms.

Description:

This enumeration lists supported hash algorithms.

4.7.3.9 CpaCySymHashNestedModeSetupData

```
typedef struct _CpaCySymHashNestedModeSetupData CpaCySymHashNestedModeSetupData
```

Hash Mode Nested Setup Data.

Description:

This structure contains data relating to a hash session in CPA_CY_SYM_HASH_MODE_NESTED mode.

4.7.3.10 CpaCySymHashAuthModeSetupData

```
typedef struct _CpaCySymHashAuthModeSetupData CpaCySymHashAuthModeSetupData
```

Hash Auth Mode Setup Data.

Description:

This structure contains data relating to a hash session in CPA_CY_SYM_HASH_MODE_AUTH mode.

4.7.3.11 CpaCySymHashSetupData

```
typedef struct _CpaCySymHashSetupData CpaCySymHashSetupData
```

Hash Setup Data.

Description:

This structure contains data relating to a hash session. The fields hashAlgorithm, hashMode and digestResultLenInBytes are common to all three hash modes and MUST be set for each mode.

4.7.3.12 CpaCySymAlgChainOrder

```
typedef enum _CpaCySymAlgChainOrder CpaCySymAlgChainOrder
```

Algorithm Chaining Operation Ordering

Description:

This enum defines the ordering of operations for algorithm chaining.

4.7.3.13 CpaCySymSessionSetupData

```
typedef struct _CpaCySymSessionSetupData CpaCySymSessionSetupData
```

Session Setup Data.

Description:

This structure contains data relating to setting up a session. The client needs to complete the information in this structure in order to setup a session.

4.7.3.14 CpaCySymSessionUpdateData

```
typedef struct _CpaCySymSessionUpdateData CpaCySymSessionUpdateData
```

Session Update Data.

Description:

This structure contains data relating to resetting a session.

4.7.3.15 CpaCySymOpData

```
typedef struct _CpaCySymOpData CpaCySymOpData
```

Cryptographic Component Operation Data.

Description:

This structure contains data relating to performing cryptographic processing on a data buffer. This request is used with cpaCySymPerformOp() call for performing cipher, hash, auth cipher or a combined hash and cipher operation.

See also

CpaCySymPacketType

Note

If the client modifies or frees the memory referenced in this structure after it has been submitted to the cpaCySymPerformOp function, and before it has been returned in the callback, undefined behavior will result.

4.7.3.16 CPA_DEPRECATED

```
typedef struct _CpaCySymStats CPA_DEPRECATED
```

Cryptographic Component Statistics.

Deprecated As of v1.3 of the cryptographic API, this structure has been deprecated, replaced by CpaCySymStats64.

Description:

This structure contains statistics on the Symmetric Cryptographic operations. Statistics are set to zero when the component is initialized.

4.7.3.17 CpaCySymStats64

```
typedef struct _CpaCySymStats64 CpaCySymStats64
```

Cryptographic Component Statistics (64-bit version).

Description:

This structure contains a 64-bit version of the statistics on the Symmetric Cryptographic operations. Statistics are set to zero when the component is initialized.

4.7.3.18 CpaCySymCbFunc

```
typedef void(* CpaCySymCbFunc) (void *pCallbackTag, CpaStatus status, const CpaCySymOp
operationType, void *pOpData, CpaBufferList *pDstBuffer, CpaBoolean verifyResult)
```

Definition of callback function

Description:

This is the callback function prototype. The callback function is registered by the application using the cpaCySymInitSession() function call.

Context:

This callback function can be executed in a context that DOES NOT permit sleeping to occur.

Assumptions:

None

Side-Effects:

None

Reentrant:

No

Thread-safe:

Yes

Parameters

in	*pCallbackTag*	Opaque value provided by user while making individual function call.
in	*status*	Status of the operation. Valid values are CPA_STATUS_SUCCESS, CPA_STATUS_FAIL and CPA_STATUS_UNSUPPORTED.
in	*operationType*	Identifies the operation type that was requested in the cpaCySymPerformOp function.
in	*pOpData*	Pointer to structure with input parameters.
in	*pDstBuffer*	Caller MUST allocate a sufficiently sized destination buffer to hold the data output. For out-of-place processing the data outside the cryptographic regions in the source buffer are copied into the destination buffer. To perform "in-place" processing set the pDstBuffer parameter in cpaCySymPerformOp function to point at the same location as pSrcBuffer. For optimum performance, the data pointed to SHOULD be 8-byte aligned.
in	*verifyResult*	This parameter is valid when the verifyDigest option is set in the CpaCySymSessionSetupData structure. A value of CPA_TRUE indicates that the compare succeeded. A value of CPA_FALSE indicates that the compare failed for an unspecified reason.

Return values

None	

Precondition

Component has been initialized.

Postcondition

None

Note

None

See also

cpaCySymInitSession(), cpaCySymRemoveSession()

Definition at line 1068 of file cpa_cy_sym.h.

4.7.3.19 CpaCySymCapabilitiesInfo

```
typedef struct _CpaCySymCapabilitiesInfo CpaCySymCapabilitiesInfo
```

Symmetric Capabilities Info

Description:

This structure contains the capabilities that vary across implementations of the symmetric sub-API of the cryptographic API. This structure is used in conjunction with cpaCySymQueryCapabilities() to determine the capabilities supported by a particular API implementation.

For example, to see if an implementation supports cipher CPA_CY_SYM_CIPHER_AES_CBC, use the code

```
if (CPA_BITMAP_BIT_TEST(capInfo.ciphers, CPA_CY_SYM_CIPHER_AES_CBC))
{
    // algo is supported
}
else
{
    // algo is not supported
}
```

```
 The client MUST allocate memory for this structure and any members
 that require memory.  When the structure is passed into the function
 ownership of the memory passes to the function. Ownership of the
 memory returns to the client when the function returns.
```

4.7.4 Enumeration Type Documentation

4.7.4.1 _CpaCySymPacketType

`enum _CpaCySymPacketType`

Packet type for the cpaCySymPerformOp function

Description:

Enumeration which is used to indicate to the symmetric cryptographic perform function on which type of packet the operation is required to be invoked. Multi-part cipher and hash operations are useful when processing needs to be performed on a message which is available to the client in multiple parts (for example due to network fragmentation of the packet).

Note

There are some restrictions regarding the operations on which partial packet processing is supported. For details, see the function cpaCySymPerformOp.

See also

cpaCySymPerformOp()

Enumerator

CPA_CY_SYM_PACKET_TYPE_FULL	Perform an operation on a full packet
CPA_CY_SYM_PACKET_TYPE_PARTIAL	Perform a partial operation and maintain the state of the partial operation within the session. This is used for either the first or subsequent packets within a partial packet flow.
CPA_CY_SYM_PACKET_TYPE_LAST_PARTIAL	Complete the last part of a multi-part operation

Definition at line 74 of file cpa_cy_sym.h.

4.7.4.2 _CpaCySymOp

`enum _CpaCySymOp`

Types of operations supported by the cpaCySymPerformOp function.

Description:

This enumeration lists different types of operations supported by the cpaCySymPerformOp function. The operation type is defined during session registration and cannot be changed for a session once it has been setup.

See also

cpaCySymPerformOp

Enumerator

CPA_CY_SYM_OP_NONE	No operation
CPA_CY_SYM_OP_CIPHER	Cipher only operation on the data
CPA_CY_SYM_OP_HASH	Hash only operation on the data
CPA_CY_SYM_OP_ALGORITHM_CHAINING	Chain any cipher with any hash operation. The order depends on the value in the CpaCySymAlgChainOrder enum. This value is also used for authenticated ciphers (GCM and CCM), in which case the cipherAlgorithm should take one of the values CPA_CY_SYM_CIPHER_AES_CCM or CPA_CY_SYM_CIPHER_AES_GCM, while the hashAlgorithm should take the corresponding value CPA_CY_SYM_HASH_AES_CCM or CPA_CY_SYM_HASH_AES_GCM.

Definition at line 98 of file cpa_cy_sym.h.

4.7.4.3 _CpaCySymCipherAlgorithm

```
enum _CpaCySymCipherAlgorithm
```

Cipher algorithms.

Description:

This enumeration lists supported cipher algorithms and modes.

Enumerator

CPA_CY_SYM_CIPHER_NULL	NULL cipher algorithm. No mode applies to the NULL algorithm.
CPA_CY_SYM_CIPHER_ARC4	(A)RC4 cipher algorithm
CPA_CY_SYM_CIPHER_AES_ECB	AES algorithm in ECB mode
CPA_CY_SYM_CIPHER_AES_CBC	AES algorithm in CBC mode
CPA_CY_SYM_CIPHER_AES_CTR	AES algorithm in Counter mode
CPA_CY_SYM_CIPHER_AES_CCM	AES algorithm in CCM mode. This authenticated cipher is only supported when the hash mode is also set to CPA_CY_SYM_HASH_MODE_AUTH. When this cipher algorithm is used the CPA_CY_SYM_HASH_AES_CCM element of the CpaCySymHashAlgorithm enum MUST be used to set up the related CpaCySymHashSetupData structure in the session context.
CPA_CY_SYM_CIPHER_AES_GCM	AES algorithm in GCM mode. This authenticated cipher is only supported when the hash mode is also set to CPA_CY_SYM_HASH_MODE_AUTH. When this cipher algorithm is used the CPA_CY_SYM_HASH_AES_GCM element of the CpaCySymHashAlgorithm enum MUST be used to set up the related CpaCySymHashSetupData structure in the session context.
CPA_CY_SYM_CIPHER_DES_ECB	DES algorithm in ECB mode

Enumerator

CPA_CY_SYM_CIPHER_DES_CBC	DES algorithm in CBC mode
CPA_CY_SYM_CIPHER_3DES_ECB	Triple DES algorithm in ECB mode
CPA_CY_SYM_CIPHER_3DES_CBC	Triple DES algorithm in CBC mode
CPA_CY_SYM_CIPHER_3DES_CTR	Triple DES algorithm in CTR mode
CPA_CY_SYM_CIPHER_KASUMI_F8	Kasumi algorithm in F8 mode
CPA_CY_SYM_CIPHER_SNOW3G_UEA2	SNOW3G algorithm in UEA2 mode
CPA_CY_SYM_CIPHER_AES_F8	AES algorithm in F8 mode
CPA_CY_SYM_CIPHER_AES_XTS	AES algorithm in XTS mode
CPA_CY_SYM_CIPHER_ZUC_EEA3	ZUC algorithm in EEA3 mode
CPA_CY_SYM_CIPHER_CHACHA	ChaCha20 Cipher Algorithm. This cipher is only supported for algorithm chaining. When selected, the hash algorithm must be set to CPA_CY_SYM_HASH_POLY and the hash mode must be set to CPA_CY_SYM_HASH_MODE_AUTH.
CPA_CY_SYM_CIPHER_SM4_ECB	SM4 algorithm in ECB mode This cipher supports 128 bit keys only and does not support partial processing.
CPA_CY_SYM_CIPHER_SM4_CBC	SM4 algorithm in CBC mode This cipher supports 128 bit keys only and does not support partial processing.
CPA_CY_SYM_CIPHER_SM4_CTR	SM4 algorithm in CTR mode This cipher supports 128 bit keys only and does not support partial processing.

Definition at line 126 of file cpa_cy_sym.h.

4.7.4.4 _CpaCySymCipherDirection

`enum _CpaCySymCipherDirection`

Symmetric Cipher Direction

Description:

This enum indicates the cipher direction (encryption or decryption).

Enumerator

CPA_CY_SYM_CIPHER_DIRECTION_ENCRYPT	Encrypt Data
CPA_CY_SYM_CIPHER_DIRECTION_DECRYPT	Decrypt Data

Definition at line 212 of file cpa_cy_sym.h.

4.7.4.5 _CpaCySymHashMode

`enum _CpaCySymHashMode`

Symmetric Hash mode

Description:

This enum indicates the Hash Mode.

Enumerator

CPA_CY_SYM_HASH_MODE_PLAIN	Plain hash. Can be specified for MD5 and the SHA family of hash algorithms.
CPA_CY_SYM_HASH_MODE_AUTH	Authenticated hash. This mode may be used in conjunction with the MD5 and SHA family of algorithms to specify HMAC. It MUST also be specified with all of the remaining algorithms, all of which are in fact authentication algorithms.
CPA_CY_SYM_HASH_MODE_NESTED	Nested hash. Can be specified for MD5 and the SHA family of hash algorithms.

Definition at line 270 of file cpa_cy_sym.h.

4.7.4.6 _CpaCySymHashAlgorithm

`enum _CpaCySymHashAlgorithm`

Hash algorithms.

Description:

This enumeration lists supported hash algorithms.

Enumerator

CPA_CY_SYM_HASH_NONE	No hash algorithm.
CPA_CY_SYM_HASH_MD5	MD5 algorithm. Supported in all 3 hash modes
CPA_CY_SYM_HASH_SHA1	128 bit SHA algorithm. Supported in all 3 hash modes
CPA_CY_SYM_HASH_SHA224	224 bit SHA algorithm. Supported in all 3 hash modes
CPA_CY_SYM_HASH_SHA256	256 bit SHA algorithm. Supported in all 3 hash modes
CPA_CY_SYM_HASH_SHA384	384 bit SHA algorithm. Supported in all 3 hash modes
CPA_CY_SYM_HASH_SHA512	512 bit SHA algorithm. Supported in all 3 hash modes
CPA_CY_SYM_HASH_AES_XCBC	AES XCBC algorithm. This is only supported in the hash mode CPA_CY_SYM_HASH_MODE_AUTH.
CPA_CY_SYM_HASH_AES_CCM	AES algorithm in CCM mode. This authenticated cipher requires that the hash mode is set to CPA_CY_SYM_HASH_MODE_AUTH. When this hash algorithm is used, the CPA_CY_SYM_CIPHER_AES_CCM element of the CpaCySymCipherAlgorithm enum MUST be used to set up the related CpaCySymCipherSetupData structure in the session context.
CPA_CY_SYM_HASH_AES_GCM	AES algorithm in GCM mode. This authenticated cipher requires that the hash mode is set to CPA_CY_SYM_HASH_MODE_AUTH. When this hash algorithm is used, the CPA_CY_SYM_CIPHER_AES_GCM element of the CpaCySymCipherAlgorithm enum MUST be used to set up the related CpaCySymCipherSetupData structure in the session context.

Enumerator

CPA_CY_SYM_HASH_KASUMI_F9	Kasumi algorithm in F9 mode. This is only supported in the hash mode CPA_CY_SYM_HASH_MODE_AUTH.
CPA_CY_SYM_HASH_SNOW3G_UIA2	SNOW3G algorithm in UIA2 mode. This is only supported in the hash mode CPA_CY_SYM_HASH_MODE_AUTH.
CPA_CY_SYM_HASH_AES_CMAC	AES CMAC algorithm. This is only supported in the hash mode CPA_CY_SYM_HASH_MODE_AUTH.
CPA_CY_SYM_HASH_AES_GMAC	AES GMAC algorithm. This is only supported in the hash mode CPA_CY_SYM_HASH_MODE_AUTH. When this hash algorithm is used, the CPA_CY_SYM_CIPHER_AES_GCM element of the CpaCySymCipherAlgorithm enum MUST be used to set up the related CpaCySymCipherSetupData structure in the session context.
CPA_CY_SYM_HASH_AES_CBC_MAC	AES-CBC-MAC algorithm. This is only supported in the hash mode CPA_CY_SYM_HASH_MODE_AUTH. Only 128-bit keys are supported.
CPA_CY_SYM_HASH_ZUC_EIA3	ZUC algorithm in EIA3 mode
CPA_CY_SYM_HASH_SHA3_256	256 bit SHA-3 algorithm. Only CPA_CY_SYM_HASH_MODE_PLAIN and CPA_CY_SYM_HASH_MODE_AUTH are supported, that is, the hash mode CPA_CY_SYM_HASH_MODE_NESTED is not supported for this algorithm. Partial requests are not supported, that is, only requests of CPA_CY_SYM_PACKET_TYPE_FULL are supported.
CPA_CY_SYM_HASH_SHA3_224	224 bit SHA-3 algorithm. Only CPA_CY_SYM_HASH_MODE_PLAIN and CPA_CY_SYM_HASH_MODE_AUTH are supported, that is, the hash mode CPA_CY_SYM_HASH_MODE_NESTED is not supported for this algorithm.
CPA_CY_SYM_HASH_SHA3_384	384 bit SHA-3 algorithm. Only CPA_CY_SYM_HASH_MODE_PLAIN and CPA_CY_SYM_HASH_MODE_AUTH are supported, that is, the hash mode CPA_CY_SYM_HASH_MODE_NESTED is not supported for this algorithm. Partial requests are not supported, that is, only requests of CPA_CY_SYM_PACKET_TYPE_FULL are supported.
CPA_CY_SYM_HASH_SHA3_512	512 bit SHA-3 algorithm. Only CPA_CY_SYM_HASH_MODE_PLAIN and CPA_CY_SYM_HASH_MODE_AUTH are supported, that is, the hash mode CPA_CY_SYM_HASH_MODE_NESTED is not supported for this algorithm. Partial requests are not supported, that is, only requests of CPA_CY_SYM_PACKET_TYPE_FULL are supported.
CPA_CY_SYM_HASH_SHAKE_128	128 bit SHAKE algorithm. This is only supported in the hash mode CPA_CY_SYM_HASH_MODE_PLAIN. Partial requests are not supported, that is, only requests of CPA_CY_SYM_PACKET_TYPE_FULL are supported.
CPA_CY_SYM_HASH_SHAKE_256	256 bit SHAKE algorithm. This is only supported in the hash mode CPA_CY_SYM_HASH_MODE_PLAIN. Partial requests are not supported, that is, only requests of CPA_CY_SYM_PACKET_TYPE_FULL are supported.

Enumerator

CPA_CY_SYM_HASH_POLY	Poly1305 hash algorithm. This is only supported in the hash mode CPA_CY_SYM_HASH_MODE_AUTH. This hash algorithm is only supported as part of an algorithm chain with AES_CY_SYM_CIPHER_CHACHA to implement the ChaCha20-Poly1305 AEAD algorithm.
CPA_CY_SYM_HASH_SM3	SM3 hash algorithm. Supported in all 3 hash modes.

Definition at line 294 of file cpa_cy_sym.h.

4.7.4.7 _CpaCySymAlgChainOrder

```
enum _CpaCySymAlgChainOrder
```

Algorithm Chaining Operation Ordering

Description:

This enum defines the ordering of operations for algorithm chaining.

Enumerator

<table>
<tr>
<td>CPA_CY_SYM_ALG_CHAIN_ORDER_HASH_TH↩
EN_CIPHER</td>
<td>Perform the hash operation followed by the cipher operation. If it is required that the result of the hash (i.e. the digest) is going to be included in the data to be ciphered, then:

• The digest MUST be placed in the destination buffer at the location corresponding to the end of the data region to be hashed (hashStartSrcOffsetInBytes + messageLenToHashInBytes), i.e. there must be no gaps between the start of the digest and the end of the data region to be hashed.

• The messageLenToCipherInBytes member of the CpaCySymOpData structure must be equal to the overall length of the plain text, the digest length and any (optional) trailing data that is to be included.

• The messageLenToCipherInBytes must be a multiple to the block size if a block cipher is being used.

The following is an example of the layout of the buffer before the operation, after the hash, and after the cipher:
<pre>
+-------------------------+----------------+
| Plaintext | Tail |
+-------------------------+----------------+
<-messageLenToHashInBytes->

+-------------------------+--------+-------+
| Plaintext | Digest | Tail |
+-------------------------+--------+-------+
<--------messageLenToCipherInBytes-------->

+--+
| Cipher Text |
+--+
</pre></td>
</tr>
<tr>
<td>CPA_CY_SYM_ALG_CHAIN_ORDER_CIPHER_T↩
HEN_HASH</td>
<td>Perform the cipher operation followed by the hash operation. The hash operation will be performed on the ciphertext resulting from the cipher operation.
The following is an example of the layout of the buffer before the operation, after the cipher, and after the hash:
<pre>
+--------+---------------------------+----------------+
| Head | Plaintext | Tail |
+--------+---------------------------+----------------+
         <-messageLenToCipherInBytes->

+--------+---------------------------+----------------+
| Head | Ciphertext | Tail |
+--------+---------------------------+----------------+
<------messageLenToHashInBytes------->

+--------+---------------------------+--------+-------+
| Head | Ciphertext | Digest | Tail |
+--------+---------------------------+--------+-------+
</pre></td>
</tr>
</table>

Definition at line 543 of file cpa_cy_sym.h.

4.7.5 Function Documentation

4.7.5.1 cpaCySymSessionCtxGetSize()

```
CpaStatus cpaCySymSessionCtxGetSize (
            const CpaInstanceHandle instanceHandle,
            const CpaCySymSessionSetupData * pSessionSetupData,
            Cpa32U * pSessionCtxSizeInBytes )
```

Gets the size required to store a session context.

Description:

This function is used by the client to determine the size of the memory it must allocate in order to store the session context. This MUST be called before the client allocates the memory for the session context and before the client calls the cpaCySymInitSession function.

For a given implementation of this API, it is safe to assume that cpaCySymSessionCtxGetSize() will always return the same size and that the size will not be different for different setup data parameters. However, it should be noted that the size may change: (1) between different implementations of the API (e.g. between software and hardware implementations or between different hardware implementations) (2) between different releases of the same API implementation.

The size returned by this function is the smallest size needed to support all possible combinations of setup data parameters. Some setup data parameter combinations may fit within a smaller session context size. The alternate cpaCySymSessionCtxGetDynamicSize() function will return the smallest size needed to fit the provided setup data parameters.

Context:

This is a synchronous function that cannot sleep. It can be executed in a context that does not permit sleeping.

Assumptions:

None

Side-Effects:

None

Blocking:

No.

Reentrant:

No

Thread-safe:

Yes

Parameters

in	*instanceHandle*	Instance handle.
in	*pSessionSetupData*	Pointer to session setup data which contains parameters which are static for a given cryptographic session such as operation type, mechanisms, and keys for cipher and/or hash operations.
out	*pSessionCtxSizeInBytes*	The amount of memory in bytes required to hold the Session Context.

Return values

CPA_STATUS_SUCCESS	Function executed successfully.
CPA_STATUS_FAIL	Function failed.
CPA_STATUS_INVALID_PARAM	Invalid parameter passed in.
CPA_STATUS_RESOURCE	Error related to system resources.
CPA_STATUS_UNSUPPORTED	Function is not supported.

Precondition

The component has been initialized via cpaCyStartInstance function.

Postcondition

None

Note

This is a synchronous function and has no completion callback associated with it.

See also

CpaCySymSessionSetupData cpaCySymInitSession() cpaCySymSessionCtxGetDynamicSize() cpaCySymPerformOp()

4.7.5.2 cpaCySymSessionCtxGetDynamicSize()

```
CpaStatus cpaCySymSessionCtxGetDynamicSize (
            const CpaInstanceHandle instanceHandle,
            const CpaCySymSessionSetupData * pSessionSetupData,
            Cpa32U * pSessionCtxSizeInBytes )
```

Gets the minimum size required to store a session context.

Description:

This function is used by the client to determine the smallest size of the memory it must allocate in order to store the session context. This MUST be called before the client allocates the memory for the session context and before the client calls the cpaCySymInitSession function.

This function is an alternate to cpaCySymSessionGetSize(). cpaCySymSessionCtxGetSize() will return a fixed size which is the minimum memory size needed to support all possible setup data parameter combinations. cpaCySymSessionCtxGetDynamicSize() will return the minimum memory size needed to support the specific session setup data parmeters provided. This size may be different for different setup data parameters.

Context:

This is a synchronous function that cannot sleep. It can be executed in a context that does not permit sleeping.

Assumptions:

None

Side-Effects:

None

Blocking:

No.

Reentrant:

No

Thread-safe:

Yes

Parameters

in	*instanceHandle*	Instance handle.
in	*pSessionSetupData*	Pointer to session setup data which contains parameters which are static for a given cryptographic session such as operation type, mechanisms, and keys for cipher and/or hash operations.
out	*pSessionCtxSizeInBytes*	The amount of memory in bytes required to hold the Session Context.

Return values

CPA_STATUS_SUCCESS	Function executed successfully.
CPA_STATUS_FAIL	Function failed.
CPA_STATUS_INVALID_PARAM	Invalid parameter passed in.
CPA_STATUS_RESOURCE	Error related to system resources.
CPA_STATUS_UNSUPPORTED	Function is not supported.

Precondition

The component has been initialized via cpaCyStartInstance function.

Postcondition

None

Note

This is a synchronous function and has no completion callback associated with it.

See also

CpaCySymSessionSetupData cpaCySymInitSession() cpaCySymSessionCtxGetSize() cpaCySymPerformOp()

4.7.5.3 cpaCySymInitSession()

```
CpaStatus cpaCySymInitSession (
            const CpaInstanceHandle instanceHandle,
            const CpaCySymCbFunc pSymCb,
            const CpaCySymSessionSetupData * pSessionSetupData,
            CpaCySymSessionCtx sessionCtx )
```

Initialize a session for symmetric cryptographic API.

Description:

This function is used by the client to initialize an asynchronous completion callback function for the symmetric cryptographic operations. Clients MAY register multiple callback functions using this function. The callback function is identified by the combination of userContext, pSymCb and session context (sessionCtx). The session context is the handle to the session and needs to be passed when processing calls. Callbacks on completion of operations within a session are guaranteed to be in the same order they were submitted in.

Context:

This is a synchronous function and it cannot sleep. It can be executed in a context that does not permit sleeping.

Assumptions:

None

Side-Effects:

None

Blocking:

No.

Reentrant:

No

Thread-safe:

Yes

Parameters

in	*instanceHandle*	Instance handle.
in	*pSymCb*	Pointer to callback function to be registered. Set to NULL if the cpaCySymPerformOp function is required to work in a synchronous manner.
in	*pSessionSetupData*	Pointer to session setup data which contains parameters which are static for a given cryptographic session such as operation type, mechanisms, and keys for cipher and/or hash operations.
out	*sessionCtx*	Pointer to the memory allocated by the client to store the session context. This will be initialized with this function. This value needs to be passed to subsequent processing calls.

Return values

CPA_STATUS_SUCCESS	Function executed successfully.
CPA_STATUS_FAIL	Function failed.
CPA_STATUS_RETRY	Resubmit the request.
CPA_STATUS_INVALID_PARAM	Invalid parameter passed in.
CPA_STATUS_RESOURCE	Error related to system resources.
CPA_STATUS_RESTARTING	API implementation is restarting. Resubmit the request.
CPA_STATUS_UNSUPPORTED	Function is not supported.

Precondition

The component has been initialized via cpaCyStartInstance function.

Postcondition

None

Note

This is a synchronous function and has no completion callback associated with it.

See also

CpaCySymSessionCtx, CpaCySymCbFunc, CpaCySymSessionSetupData, cpaCySymRemoveSession(), cpaCySymPerformOp()

4.7.5.4 cpaCySymRemoveSession()

```
CpaStatus cpaCySymRemoveSession (
            const CpaInstanceHandle instanceHandle,
            CpaCySymSessionCtx pSessionCtx )
```

Remove (delete) a symmetric cryptographic session.

Description:

This function will remove a previously initialized session context and the installed callback handler function. Removal will fail if outstanding calls still exist for the initialized session handle. The client needs to retry the remove function at a later time. The memory for the session context MUST not be freed until this call has completed successfully.

Context:

This is a synchronous function that cannot sleep. It can be executed in a context that does not permit sleeping.

Assumptions:

None

Side-Effects:

None

Blocking:

No.

Reentrant:

No

Thread-safe:

Yes

Parameters

`in`	*instanceHandle*	Instance handle.
`in,out`	*pSessionCtx*	Session context to be removed.

Return values

CPA_STATUS_SUCCESS	Function executed successfully.
CPA_STATUS_FAIL	Function failed.
CPA_STATUS_RETRY	Resubmit the request.
CPA_STATUS_INVALID_PARAM	Invalid parameter passed in.
CPA_STATUS_RESOURCE	Error related to system resources.
CPA_STATUS_RESTARTING	API implementation is restarting. Resubmit the request.
CPA_STATUS_UNSUPPORTED	Function is not supported.

Precondition

The component has been initialized via cpaCyStartInstance function.

Postcondition

None

Note

Note that this is a synchronous function and has no completion callback associated with it.

See also

CpaCySymSessionCtx, cpaCySymInitSession()

4.7.5.5 cpaCySymUpdateSession()

```
CpaStatus cpaCySymUpdateSession (
            CpaCySymSessionCtx sessionCtx,
            const CpaCySymSessionUpdateData * pSessionUpdateData )
```

Update a session.

Description:

This function is used to update certain parameters of a session, as specified by the CpaCySymSessionUpdateData data structure.

It can be used on sessions created with either the so-called Traditional API (cpaCySymInitSession) or the Data Plane API (cpaCySymDpInitSession).

In order for this function to operate correctly, two criteria must be met:

- In the case of sessions created with the Traditional API, the session must be stateless, i.e. the field partialsNotRequired of the CpaCySymSessionSetupData data structure must be FALSE. (Sessions created using the Data Plane API are always stateless.)
- There must be no outstanding requests in flight for the session. The application can call the function cpaCySymSessionInUse to test for this.

  Note that in the case of multi-threaded applications (which are supported using the Traditional API only), this function may fail even if a previous invocation of the function cpaCySymSessionInUse indicated that there were no outstanding requests.

Parameters

in	*sessionCtx*	Identifies the session to be reset.
in	*pSessionUpdateData*	Pointer to session data which contains the parameters to be updated.

Return values

CPA_STATUS_SUCCESS	Function executed successfully.
CPA_STATUS_FAIL	Function failed.
CPA_STATUS_RETRY	Resubmit the request.
CPA_STATUS_INVALID_PARAM	Invalid parameter passed in.
CPA_STATUS_RESOURCE	Error related to system resources.
CPA_STATUS_RESTARTING	API implementation is restarting. Resubmit the request.
CPA_STATUS_UNSUPPORTED	Function is not supported.

Precondition

The component has been initialized via cpaCyStartInstance function.

Postcondition

None

Note

This is a synchronous function and has no completion callback associated with it.

4.7.5.6 cpaCySymSessionInUse()

```
CpaStatus cpaCySymSessionInUse (
            CpaCySymSessionCtx sessionCtx,
            CpaBoolean * pSessionInUse )
```

Indicates whether there are outstanding requests on a given session.

Description:

This function is used to test whether there are outstanding requests in flight for a specified session. This may be used before resetting session parameters using the function cpaCySymResetSession. See some additional notes on multi-threaded applications described on that function.

Parameters

in	*sessionCtx*	Identifies the session to be reset.
out	*pSessionInUse*	Returns CPA_TRUE if there are outstanding requests on the session, or CPA_FALSE otherwise.

4.7.5.7 cpaCySymPerformOp()

```
CpaStatus cpaCySymPerformOp (
```

```
const CpaInstanceHandle instanceHandle,
void * pCallbackTag,
const CpaCySymOpData * pOpData,
const CpaBufferList * pSrcBuffer,
CpaBufferList * pDstBuffer,
CpaBoolean * pVerifyResult )
```

Perform a symmetric cryptographic operation on an existing session.

Description:

Performs a cipher, hash or combined (cipher and hash) operation on the source data buffer using supported symmetric key algorithms and modes.

This function maintains cryptographic state between calls for partial cryptographic operations. If a partial cryptographic operation is being performed, then on a per-session basis, the next part of the multi-part message can be submitted prior to previous parts being completed, the only limitation being that all parts must be performed in sequential order.

If for any reason a client wishes to terminate the partial packet processing on the session (for example if a packet fragment was lost) then the client MUST remove the session.

When using partial packet processing with algorithm chaining, only the cipher state is maintained between calls. The hash state is not be maintained between calls. Instead the hash digest will be generated/verified for each call. If both the cipher state and hash state need to be maintained between calls, algorithm chaining cannot be used.

The following restrictions apply to the length:

- When performing block based operations on a partial packet (excluding the final partial packet), the data that is to be operated on MUST be a multiple of the block size of the algorithm being used. This restriction only applies to the cipher state when using partial packets with algorithm chaining.
- The final block must not be of length zero (0) if the operation being performed is the authentication algorithm CPA_CY_SYM_HASH_AES_XCBC. This is because this algorithm requires that the final block be XORed with another value internally. If the length is zero, then the return code CPA_STATUS_INVALID_PARAM will be returned.
- The length of the final block must be greater than or equal to 16 bytes when using the CPA_CY_SYM_CIPHER_AES_XTS cipher algorithm.

Partial packet processing is supported only when the following conditions are true:

- The cipher, hash or authentication operation is "in place" (that is, pDstBuffer == pSrcBuffer)
- The cipher or hash algorithm is NOT one of Kasumi or SNOW3G
- The cipher mode is NOT F8 mode.
- The hash algorithm is NOT SHAKE
- The cipher algorithm is not SM4
- The cipher algorithm is not CPA_CY_SYM_CIPHER_CHACHA and the hash algorithm is not CPA_CY_SYM_HASH_POLY.
- The cipher algorithm is not CPA_CY_SYM_CIPHER_AES_GCM and the hash algorithm is not CPA_CY_SYM_HASH_AES_GCM.
- The instance/implementation supports partial packets as one of its capabilities (see CpaCySymCapabilitiesInfo).

The term "in-place" means that the result of the cryptographic operation is written into the source buffer. The term "out-of-place" means that the result of the cryptographic operation is written into the destination buffer. To perform "in-place" processing, set the pDstBuffer parameter to point at the same location as the pSrcBuffer parameter.

Context:

When called as an asynchronous function it cannot sleep. It can be executed in a context that does not permit sleeping. When called as a synchronous function it may sleep. It MUST NOT be executed in a context that DOES NOT permit sleeping.

Assumptions:

None

Side-Effects:

None

Blocking:

Yes when configured to operate in synchronous mode.

Reentrant:

No

Thread-safe:

Yes

Parameters

`in`	*instanceHandle*	Instance handle.
`in`	*pCallbackTag*	Opaque data that will be returned to the client in the callback.
`in`	*pOpData*	Pointer to a structure containing request parameters. The client code allocates the memory for this structure. This component takes ownership of the memory until it is returned in the callback.
`in`	*pSrcBuffer*	The source buffer. The caller MUST allocate the source buffer and populate it with data. For optimum performance, the data pointed to SHOULD be 8-byte aligned. For block ciphers, the data passed in MUST be a multiple of the relevant block size. i.e. padding WILL NOT be applied to the data. For optimum performance, the buffer should only contain the data region that the cryptographic operation(s) must be performed on. Any additional data in the source buffer may be copied to the destination buffer and this copy may degrade performance.
`out`	*pDstBuffer*	The destination buffer. The caller MUST allocate a sufficiently sized destination buffer to hold the data output (including the authentication tag in the case of CCM). Furthermore, the destination buffer must be the same size as the source buffer (i.e. the sum of lengths of the buffers in the buffer list must be the same). This effectively means that the source buffer must in fact be big enough to hold the output data, too. This is because, for out-of-place processing, the data outside the regions in the source buffer on which cryptographic operations are performed are copied into the destination buffer. To perform "in-place" processing set the pDstBuffer parameter in cpaCySymPerformOp function to point at the same location as pSrcBuffer. For optimum performance, the data pointed to SHOULD be 8-byte aligned.
`out`	*pVerifyResult*	In synchronous mode, this parameter is returned when the verifyDigest option is set in the CpaCySymSessionSetupData structure. A value of CPA_TRUE indicates that the compare succeeded. A value of CPA_FALSE indicates that the compare failed for an unspecified reason.

Return values

CPA_STATUS_SUCCESS	Function executed successfully.
CPA_STATUS_FAIL	Function failed.
CPA_STATUS_RETRY	Resubmit the request.
CPA_STATUS_INVALID_PARAM	Invalid parameter passed in.
CPA_STATUS_RESOURCE	Error related to system resource.
CPA_STATUS_RESTARTING	API implementation is restarting. Resubmit the request.
CPA_STATUS_UNSUPPORTED	Function is not supported.

Precondition

The component has been initialized via cpaCyStartInstance function. A Cryptographic session has been previously setup using the cpaCySymInitSession function call.

Postcondition

None

Note

When in asynchronous mode, a callback of type CpaCySymCbFunc is generated in response to this function call. Any errors generated during processing are reported as part of the callback status code.

See also

CpaCySymOpData, cpaCySymInitSession(), cpaCySymRemoveSession()

4.7.5.8 cpaCySymQueryStats()

```
CpaStatus CPA_DEPRECATED cpaCySymQueryStats (
            const CpaInstanceHandle instanceHandle,
            struct _CpaCySymStats * pSymStats )
```

Query symmetric cryptographic statistics for a specific instance.

Deprecated As of v1.3 of the cryptographic API, this function has been deprecated, replaced by cpaCySymQueryStats64().

Description:

This function will query a specific instance for statistics. The user MUST allocate the CpaCySymStats structure and pass the reference to that into this function call. This function will write the statistic results into the passed in CpaCySymStats structure.

Note: statistics returned by this function do not interrupt current data processing and as such can be slightly out of sync with operations that are in progress during the statistics retrieval process.

Context:

This is a synchronous function and it can sleep. It MUST NOT be executed in a context that DOES NOT permit sleeping.

Assumptions:

None

Side-Effects:

None

Blocking:

Yes

Reentrant:

No

Thread-safe:

Yes

Parameters

in	*instanceHandle*	Instance handle.
out	*pSymStats*	Pointer to memory into which the statistics will be written.

Return values

CPA_STATUS_SUCCESS	Function executed successfully.
CPA_STATUS_FAIL	Function failed.
CPA_STATUS_INVALID_PARAM	Invalid parameter passed in.
CPA_STATUS_RESOURCE	Error related to system resources.
CPA_STATUS_RESTARTING	API implementation is restarting. Resubmit the request.
CPA_STATUS_UNSUPPORTED	Function is not supported.

Precondition

Component has been initialized.

Postcondition

None

Note

This function operates in a synchronous manner, i.e. no asynchronous callback will be generated.

See also

CpaCySymStats

4.7.5.9 cpaCySymQueryStats64()

```
CpaStatus cpaCySymQueryStats64 (
            const CpaInstanceHandle instanceHandle,
            CpaCySymStats64 * pSymStats )
```

Query symmetric cryptographic statistics (64-bit version) for a specific instance.

Description:

This function will query a specific instance for statistics. The user MUST allocate the CpaCySymStats64 structure and pass the reference to that into this function call. This function will write the statistic results into the passed in CpaCySymStats64 structure.

Note: statistics returned by this function do not interrupt current data processing and as such can be slightly out of sync with operations that are in progress during the statistics retrieval process.

Context:

This is a synchronous function and it can sleep. It MUST NOT be executed in a context that DOES NOT permit sleeping.

Assumptions:

None

Side-Effects:

None

Blocking:

Yes

Reentrant:

No

Thread-safe:

Yes

Parameters

in	*instanceHandle*	Instance handle.
out	*pSymStats*	Pointer to memory into which the statistics will be written.

Return values

CPA_STATUS_SUCCESS	Function executed successfully.
CPA_STATUS_FAIL	Function failed.
CPA_STATUS_INVALID_PARAM	Invalid parameter passed in.
CPA_STATUS_RESOURCE	Error related to system resources.
CPA_STATUS_RESTARTING	API implementation is restarting. Resubmit the request.
CPA_STATUS_UNSUPPORTED	Function is not supported.

Precondition

Component has been initialized.

Postcondition

None

Note

This function operates in a synchronous manner, i.e. no asynchronous callback will be generated.

See also

CpaCySymStats64

4.7.5.10 cpaCySymQueryCapabilities()

```
CpaStatus cpaCySymQueryCapabilities (
            const CpaInstanceHandle instanceHandle,
            CpaCySymCapabilitiesInfo * pCapInfo )
```

Returns capabilities of the symmetric API group of a Cryptographic API instance.

Description:

This function is used to determine which specific capabilities are supported within the symmetric sub-group of the Cryptographic API.

Context:

The function shall not be called in an interrupt context.

Assumptions:

None

Side-Effects:

None

Blocking:

This function is synchronous and blocking.

Reentrant:

No

Thread-safe:

Yes

Parameters

in	*instanceHandle*	Handle to an instance of this API.
out	*pCapInfo*	Pointer to capabilities info structure. All fields in the structure are populated by the API instance.

Return values

CPA_STATUS_SUCCESS	Function executed successfully.
CPA_STATUS_FAIL	Function failed.
CPA_STATUS_INVALID_PARAM	Invalid parameter passed in.
CPA_STATUS_UNSUPPORTED	Function is not supported.

Precondition

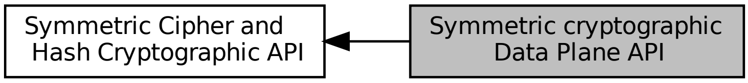
The instance has been initialized via the cpaCyStartInstance function.

Postcondition

None

4.8 Symmetric cryptographic Data Plane API

Collaboration diagram for Symmetric cryptographic Data Plane API:



Data Structures

- struct _CpaCySymDpOpData

Typedefs

- typedef void ∗ CpaCySymDpSessionCtx
- typedef struct _CpaCySymDpOpData CpaCySymDpOpData
- typedef void(∗ CpaCySymDpCbFunc) (CpaCySymDpOpData ∗pOpData, CpaStatus status, CpaBoolean verifyResult)

Functions

- CpaStatus cpaCySymDpRegCbFunc (const CpaInstanceHandle instanceHandle, const CpaCySymDpCbFunc pSymNewCb)
- CpaStatus cpaCySymDpSessionCtxGetSize (const CpaInstanceHandle instanceHandle, const CpaCySymSessionSetupData ∗pSessionSetupData, Cpa32U ∗pSessionCtxSizeInBytes)
- CpaStatus cpaCySymDpSessionCtxGetDynamicSize (const CpaInstanceHandle instanceHandle, const CpaCySymSessionSetupData ∗pSessionSetupData, Cpa32U ∗pSessionCtxSizeInBytes)
- CpaStatus cpaCySymDpInitSession (CpaInstanceHandle instanceHandle, const CpaCySymSessionSetupData ∗pSessionSetupData, CpaCySymDpSessionCtx sessionCtx)
- CpaStatus cpaCySymDpRemoveSession (const CpaInstanceHandle instanceHandle, CpaCySymDpSessionCtx sessionCtx)
- CpaStatus cpaCySymDpEnqueueOp (CpaCySymDpOpData ∗pOpData, const CpaBoolean performOpNow)
- CpaStatus cpaCySymDpEnqueueOpBatch (const Cpa32U numberRequests, CpaCySymDpOpData ∗pOpData[ ], const CpaBoolean performOpNow)
- CpaStatus cpaCySymDpPerformOpNow (CpaInstanceHandle instanceHandle)

4.8.1 Detailed Description

File: cpa_cy_sym_dp.h

Description:

These data structures and functions specify the Data Plane API for symmetric cipher, hash, and combined cipher and hash operations.

This API is recommended for data plane applications, in which the cost of offload - that is, the cycles consumed by the driver in sending requests to the hardware, and processing responses - needs to be minimized. In particular, use of this API is recommended if the following constraints are acceptable to your application:

- Thread safety is not guaranteed. Each software thread should have access to its own unique instance (CpaInstanceHandle) to avoid contention.
- Polling is used, rather than interrupts (which are expensive). Implementations of this API will provide a function (not defined as part of this API) to read responses from the hardware response queue and dispatch callback functions, as specified on this API.
- Buffers and buffer lists are passed using physical addresses, to avoid virtual to physical address translation costs.
- For GCM and CCM modes of AES, when performing decryption and verification, if verification fails, then the message buffer will NOT be zeroed. (This is a consequence of using physical addresses for the buffers.)
- The ability to enqueue one or more requests without submitting them to the hardware allows for certain costs to be amortized across multiple requests.
- Only asynchronous invocation is supported.
- There is no support for partial packets.
- Implementations may provide certain features as optional at build time, such as atomic counters.
- The "default" instance (CPA_INSTANCE_HANDLE_SINGLE) is not supported on this API. The specific handle should be obtained using the instance discovery functions (cpaCyGetNumInstances, cpaCyGetInstances).

Note

Performance Trade-Offs Different implementations of this API may have different performance trade-offs; please refer to the documentation for your implementation for details. However, the following concepts informed the definition of this API.

The API distinguishes between *enqueuing* a request and actually *submitting* that request to the cryptographic acceleration engine to be performed. This allows multiple requests to be enqueued (either individually or in batch), and then for all enqueued requests to be submitted in a single operation. The rationale is that in some (especially hardware-based) implementations, the submit operation is expensive; for example, it may incur an MMIO instruction. The API allows this cost to be amortized over a number of requests. The precise number of such requests can be tuned for optimal performance.

Specifically:

- The function cpaCySymDpEnqueueOp allows one request to be enqueued, and optionally for that request (and all previously enqueued requests) to be submitted.
- The function cpaCySymDpEnqueueOpBatch allows multiple requests to be enqueued, and optionally for those requests (and all previously enqueued requests) to be submitted.
- The function cpaCySymDpPerformOpNow enqueues no requests, but submits all previously enqueued requests.

4.8.2 Typedef Documentation

4.8.2.1 CpaCySymDpSessionCtx

```
typedef void* CpaCySymDpSessionCtx
```

Cryptographic component symmetric session context handle for the data plane API.

Description:

Handle to a cryptographic data plane session context. The memory for this handle is allocated by the client. The size of the memory that the client needs to allocate is determined by a call to the cpaCySymDpSessionCtxGetSize or cpaCySymDpSessionCtxGetDynamicSize functions. The session context memory is initialized with a call to the cpaCySymInitSession function. This memory MUST not be freed until a call to cpaCySymDpRemoveSession has completed successfully.

Definition at line 112 of file cpa_cy_sym_dp.h.

4.8.2.2 CpaCySymDpOpData

```
typedef struct _CpaCySymDpOpData CpaCySymDpOpData
```

Operation Data for cryptographic data plane API.

Description:

This structure contains data relating to a request to perform symmetric cryptographic processing on one or more data buffers.

The physical memory to which this structure points needs to be at least 8-byte aligned.

All reserved fields SHOULD NOT be written or read by the calling code.

See also

cpaCySymDpEnqueueOp, cpaCySymDpEnqueueOpBatch

4.8.2.3 CpaCySymDpCbFunc

```
typedef void(* CpaCySymDpCbFunc) (CpaCySymDpOpData *pOpData, CpaStatus status, CpaBoolean
verifyResult)
```

Definition of callback function for cryptographic data plane API.

Description:

This is the callback function prototype. The callback function is registered by the application using the cpaCySymDpRegCbFunc function call, and called back on completion of asycnhronous requests made via calls to cpaCySymDpEnqueueOp or cpaCySymDpEnqueueOpBatch.

Context:

This callback function can be executed in a context that DOES NOT permit sleeping to occur.

Assumptions:

None

Side-Effects:

None

Reentrant:

No

Thread-safe:

No

Parameters

in	*pOpData*	Pointer to the CpaCySymDpOpData object which was supplied as part of the original request.
in	*status*	Status of the operation. Valid values are CPA_STATUS_SUCCESS, CPA_STATUS_FAIL and CPA_STATUS_UNSUPPORTED.
in	*verifyResult*	This parameter is valid when the verifyDigest option is set in the CpaCySymSessionSetupData structure. A value of CPA_TRUE indicates that the compare succeeded. A value of CPA_FALSE indicates that the compare failed.

Returns

None

Precondition

Component has been initialized. Callback has been registered with cpaCySymDpRegCbFunc.

Postcondition

None

Note

None

See also

cpaCySymDpRegCbFunc

Definition at line 388 of file cpa_cy_sym_dp.h.

4.8.3 Function Documentation

4.8.3.1 cpaCySymDpRegCbFunc()

```
CpaStatus cpaCySymDpRegCbFunc (
            const CpaInstanceHandle instanceHandle,
            const CpaCySymDpCbFunc pSymNewCb )
```

Registration of the operation completion callback function.

Description:

This function allows a completion callback function to be registered. The registered callback function is invoked on completion of asycnhronous requests made via calls to cpaCySymDpEnqueueOp or cpaCySymDpEnqueueOpBatch.

If a callback function was previously registered, it is overwritten.

Context:

This is a synchronous function and it cannot sleep. It can be executed in a context that does not permit sleeping.

Assumptions:

None

Side-Effects:

None

Reentrant:

No

Thread-safe:

No

Parameters

in	*instanceHandle*	Instance on which the callback function is to be registered.
in	*pSymNewCb*	Callback function for this instance.

Return values

CPA_STATUS_SUCCESS	Function executed successfully.
CPA_STATUS_FAIL	Function failed.
CPA_STATUS_RESTARTING	API implementation is restarting. Resubmit the request.
CPA_STATUS_UNSUPPORTED	Function is not supported.

Precondition

Component has been initialized.

Postcondition

None

Note

None

See also

CpaCySymDpCbFunc

4.8.3.2 cpaCySymDpSessionCtxGetSize()

```
CpaStatus cpaCySymDpSessionCtxGetSize (
            const CpaInstanceHandle instanceHandle,
            const CpaCySymSessionSetupData * pSessionSetupData,
            Cpa32U * pSessionCtxSizeInBytes )
```

Gets the size required to store a session context for the data plane API.

Description:

This function is used by the client to determine the size of the memory it must allocate in order to store the session context. This MUST be called before the client allocates the memory for the session context and before the client calls the cpaCySymDpInitSession function.

For a given implementation of this API, it is safe to assume that cpaCySymDpSessionCtxGetSize() will always return the same size and that the size will not be different for different setup data parameters. However, it should be noted that the size may change: (1) between different implementations of the API (e.g. between software and hardware implementations or between different hardware implementations) (2) between different releases of the same API implementation.

The size returned by this function is the smallest size needed to support all possible combinations of setup data parameters. Some setup data parameter combinations may fit within a smaller session context size. The alternate cpaCySymDpSessionCtxGetDynamicSize() function will return the smallest size needed to fit the provided setup data parameters.

Context:

This is a synchronous function that cannot sleep. It can be executed in a context that does not permit sleeping.

Assumptions:

None

Side-Effects:

None

Blocking:

No

Reentrant:

No

Thread-safe:

Yes

Parameters

in	*instanceHandle*	Instance handle.
in	*pSessionSetupData*	Pointer to session setup data which contains parameters which are static for a given cryptographic session such as operation type, mechanisms, and keys for cipher and/or hash operations.
out	*pSessionCtxSizeInBytes*	The amount of memory in bytes required to hold the Session Context.

Return values

CPA_STATUS_SUCCESS	Function executed successfully.
CPA_STATUS_FAIL	Function failed.
CPA_STATUS_INVALID_PARAM	Invalid parameter passed in.
CPA_STATUS_RESOURCE	Error related to system resources.
CPA_STATUS_UNSUPPORTED	Function is not supported.

Precondition

The component has been initialized.

Postcondition

None

Note

This is a synchronous function and has no completion callback associated with it.

See also

CpaCySymSessionSetupData cpaCySymDpSessionCtxGetDynamicSize() cpaCySymDpInitSession()

4.8.3.3 cpaCySymDpSessionCtxGetDynamicSize()

```
CpaStatus cpaCySymDpSessionCtxGetDynamicSize (
            const CpaInstanceHandle instanceHandle,
            const CpaCySymSessionSetupData * pSessionSetupData,
            Cpa32U * pSessionCtxSizeInBytes )
```

Gets the minimum size required to store a session context for the data plane API.

Description:

This function is used by the client to determine the smallest size of the memory it must allocate in order to store the session context. This MUST be called before the client allocates the memory for the session context and before the client calls the cpaCySymDpInitSession function.

This function is an alternate to cpaCySymDpSessionGetSize(). cpaCySymDpSessionCtxGetSize() will return a fixed size which is the minimum memory size needed to support all possible setup data parameter combinations. cpaCySymDpSessionCtxGetDynamicSize() will return the minimum memory size needed to support the specific session setup data parmeters provided. This size may be different for different setup data parameters.

Context:

This is a synchronous function that cannot sleep. It can be executed in a context that does not permit sleeping.

Assumptions:

None

Side-Effects:

None

Blocking:

No

Reentrant:

No

Thread-safe:

Yes

Parameters

in	*instanceHandle*	Instance handle.
in	*pSessionSetupData*	Pointer to session setup data which contains parameters which are static for a given cryptographic session such as operation type, mechanisms, and keys for cipher and/or hash operations.
out	*pSessionCtxSizeInBytes*	The amount of memory in bytes required to hold the Session Context.

Return values

CPA_STATUS_SUCCESS	Function executed successfully.
CPA_STATUS_FAIL	Function failed.
CPA_STATUS_INVALID_PARAM	Invalid parameter passed in.
CPA_STATUS_RESOURCE	Error related to system resources.
CPA_STATUS_UNSUPPORTED	Function is not supported.

Precondition

The component has been initialized.

Postcondition

None

Note

This is a synchronous function and has no completion callback associated with it.

See also

CpaCySymSessionSetupData cpaCySymDpSessionCtxGetSize() cpaCySymDpInitSession()

4.8.3.4 cpaCySymDpInitSession()

```
CpaStatus cpaCySymDpInitSession (
            CpaInstanceHandle instanceHandle,
            const CpaCySymSessionSetupData * pSessionSetupData,
            CpaCySymDpSessionCtx sessionCtx )
```

Initialize a session for the symmetric cryptographic data plane API.

Description:

This function is used by the client to initialize an asynchronous session context for symmetric cryptographic data plane operations. The returned session context is the handle to the session and needs to be passed when requesting cryptographic operations to be performed.

Only sessions created using this function may be used when invoking functions on this API

The session can be removed using cpaCySymDpRemoveSession.

Context:

This is a synchronous function and it cannot sleep. It can be executed in a context that does not permit sleeping.

Assumptions:

None

Side-Effects:

None

Blocking:

No

Reentrant:

No

Thread-safe:

No

Parameters

in	*instanceHandle*	Instance to which the requests will be submitted.
in	*pSessionSetupData*	Pointer to session setup data which contains parameters that are static for a given cryptographic session such as operation type, algorithm, and keys for cipher and/or hash operations.
out	*sessionCtx*	Pointer to the memory allocated by the client to store the session context. This memory must be physically contiguous, and its length (in bytes) must be at least as big as specified by a call to cpaCySymDpSessionCtxGetSize. This memory will be initialized with this function. This value needs to be passed to subsequent processing calls.

Return values

CPA_STATUS_SUCCESS	Function executed successfully.
CPA_STATUS_FAIL	Function failed.
CPA_STATUS_RETRY	Resubmit the request.
CPA_STATUS_INVALID_PARAM	Invalid parameter passed in.
CPA_STATUS_RESOURCE	Error related to system resources.
CPA_STATUS_RESTARTING	API implementation is restarting. Resubmit the request.
CPA_STATUS_UNSUPPORTED	Function is not supported.

Precondition

The component has been initialized.

Postcondition

None

Note

This is a synchronous function and has no completion callback associated with it.

See also

cpaCySymDpSessionCtxGetSize, cpaCySymDpRemoveSession

4.8.3.5 cpaCySymDpRemoveSession()

```
CpaStatus cpaCySymDpRemoveSession (
            const CpaInstanceHandle instanceHandle,
            CpaCySymDpSessionCtx sessionCtx )
```

Remove (delete) a symmetric cryptographic session for the data plane API.

Description:

This function will remove a previously initialized session context and the installed callback handler function. Removal will fail if outstanding calls still exist for the initialized session handle. The client needs to retry the remove function at a later time. The memory for the session context MUST not be freed until this call has completed successfully.

Context:

This is a synchronous function that cannot sleep. It can be executed in a context that does not permit sleeping.

Assumptions:

None

Side-Effects:

None

Blocking:

No

Reentrant:

No

Thread-safe:

No

Parameters

in	*instanceHandle*	Instance handle.
in,out	*sessionCtx*	Session context to be removed.

Return values

CPA_STATUS_SUCCESS	Function executed successfully.
CPA_STATUS_FAIL	Function failed.
CPA_STATUS_RETRY	Resubmit the request.
CPA_STATUS_INVALID_PARAM	Invalid parameter passed in.
CPA_STATUS_RESOURCE	Error related to system resources.
CPA_STATUS_RESTARTING	API implementation is restarting. Resubmit the request.
CPA_STATUS_UNSUPPORTED	Function is not supported.

Precondition

The component has been initialized.

Postcondition

None

Note

Note that this is a synchronous function and has no completion callback associated with it.

See also

CpaCySymDpSessionCtx, cpaCySymDpInitSession()

4.8.3.6 cpaCySymDpEnqueueOp()

```
CpaStatus cpaCySymDpEnqueueOp (
            CpaCySymDpOpData * pOpData,
            const CpaBoolean performOpNow )
```

Enqueue a single symmetric cryptographic request.

Description:

This function enqueues a single request to perform a cipher, hash or combined (cipher and hash) operation. Optionally, the request is also submitted to the cryptographic engine to be performed.

See note about performance trade-offs on the Symmetric cryptographic Data Plane API API.

The function is asynchronous; control is returned to the user once the request has been submitted. On completion of the request, the application may poll for responses, which will cause a callback function (registered via cpaCySymDpRegCbFunc) to be invoked. Callbacks within a session are guaranteed to be in the same order in which they were submitted.

The following restrictions apply to the pOpData parameter:

- The memory MUST be aligned on an 8-byte boundary.
- The structure MUST reside in physically contiguous memory.
- The reserved fields of the structure SHOULD NOT be written or read by the calling code.

Context:

This function will not sleep, and hence can be executed in a context that does not permit sleeping.

Side-Effects:

None

Blocking:

No

Reentrant:

No

Thread-safe:

No

Parameters

in	*pOpData*	Pointer to a structure containing the request parameters. The client code allocates the memory for this structure. This component takes ownership of the memory until it is returned in the callback, which was registered on the instance via cpaCySymDpRegCbFunc. See the above Description for restrictions that apply to this parameter.
in	*performOpNow*	Flag to specify whether the operation should be performed immediately (CPA_TRUE), or simply enqueued to be performed later (CPA_FALSE). In the latter case, the request is submitted to be performed either by calling this function again with this flag set to CPA_TRUE, or by invoking the function cpaCySymDpPerformOpNow.

Return values

CPA_STATUS_SUCCESS	Function executed successfully.
CPA_STATUS_FAIL	Function failed.
CPA_STATUS_RETRY	Resubmit the request.
CPA_STATUS_INVALID_PARAM	Invalid parameter passed in.
CPA_STATUS_RESTARTING	API implementation is restarting. Resubmit the request.
CPA_STATUS_UNSUPPORTED	Function is not supported.

Precondition

The session identified by pOpData->sessionCtx was setup using cpaCySymDpInitSession. The instance identified by pOpData->instanceHandle has had a callback function registered via cpaCySymDpRegCbFunc.

Postcondition

None

Note

A callback of type CpaCySymDpCbFunc is generated in response to this function call. Any errors generated during processing are reported as part of the callback status code.

See also

cpaCySymDpInitSession, cpaCySymDpPerformOpNow

4.8.3.7 cpaCySymDpEnqueueOpBatch()

```
CpaStatus cpaCySymDpEnqueueOpBatch (
            const Cpa32U numberRequests,
            CpaCySymDpOpData * pOpData[],
            const CpaBoolean performOpNow )
```

Enqueue multiple requests to the symmetric cryptographic data plane API.

Description:

This function enqueues multiple requests to perform cipher, hash or combined (cipher and hash) operations.

See note about performance trade-offs on the Symmetric cryptographic Data Plane API API.

The function is asynchronous; control is returned to the user once the request has been submitted. On completion of the request, the application may poll for responses, which will cause a callback function (registered via cpaCySymDpRegCbFunc) to be invoked. Separate callbacks will be invoked for each request. Callbacks within a session are guaranteed to be in the same order in which they were submitted.

The following restrictions apply to each element of the pOpData array:

- The memory MUST be aligned on an 8-byte boundary.
- The structure MUST reside in physically contiguous memory.
- The reserved fields of the structure SHOULD NOT be written or read by the calling code.

Context:

This function will not sleep, and hence can be executed in a context that does not permit sleeping.

Assumptions:

Client MUST allocate the request parameters to 8 byte alignment. Reserved elements of the CpaCySymDpOpData structure MUST be 0. The CpaCySymDpOpData structure MUST reside in physically contiguous memory.

Side-Effects:

None

Blocking:

No

Reentrant:

No

Thread-safe:

No

Parameters

in	*numberRequests*	The number of requests in the array of CpaCySymDpOpData structures.
in	*pOpData*	An array of pointers to CpaCySymDpOpData structures. Each of the CpaCySymDpOpData structure contains the request parameters for that request. The client code allocates the memory for this structure. This component takes ownership of the memory until it is returned in the callback, which was registered on the instance via cpaCySymDpRegCbFunc. See the above Description for restrictions that apply to this parameter.
in	*performOpNow*	Flag to specify whether the operation should be performed immediately (CPA_TRUE), or simply enqueued to be performed later (CPA_FALSE). In the latter case, the request is submitted to be performed either by calling this function again with this flag set to CPA_TRUE, or by invoking the function cpaCySymDpPerformOpNow.

Return values

CPA_STATUS_SUCCESS	Function executed successfully.
CPA_STATUS_FAIL	Function failed.
CPA_STATUS_RETRY	Resubmit the request.
CPA_STATUS_INVALID_PARAM	Invalid parameter passed in.
CPA_STATUS_RESTARTING	API implementation is restarting. Resubmit the request.
CPA_STATUS_UNSUPPORTED	Function is not supported.

Precondition

The session identified by pOpData[i]->sessionCtx was setup using cpaCySymDpInitSession. The instance identified by pOpData->instanceHandle[i] has had a callback function registered via cpaCySymDpRegCbFunc.

Postcondition

None

Note

Multiple callbacks of type CpaCySymDpCbFunc are generated in response to this function call (one per request). Any errors generated during processing are reported as part of the callback status code.

See also

cpaCySymDpInitSession, cpaCySymDpEnqueueOp

4.8.3.8 cpaCySymDpPerformOpNow()

```
CpaStatus cpaCySymDpPerformOpNow (
            CpaInstanceHandle instanceHandle )
```

Submit any previously enqueued requests to be performed now on the symmetric cryptographic data plane API.

Description:

If any requests/operations were enqueued via calls to cpaCySymDpEnqueueOp and/or cpaCySymDpEnqueueOpBatch, but with the flag performOpNow set to CPA_FALSE, then these operations will now be submitted to the accelerator to be performed.

See note about performance trade-offs on the Symmetric cryptographic Data Plane API API.

Context:

Will not sleep. It can be executed in a context that does not permit sleeping.

Side-Effects:

None

Blocking:

No

Reentrant:

No

Thread-safe:

No

Parameters

in	*instanceHandle*	Instance to which the requests will be submitted.

Return values

CPA_STATUS_SUCCESS	Function executed successfully.
CPA_STATUS_FAIL	Function failed.
CPA_STATUS_RETRY	Resubmit the request.
CPA_STATUS_INVALID_PARAM	Invalid parameter passed in.
CPA_STATUS_RESTARTING	API implementation is restarting. Resubmit the request.
CPA_STATUS_UNSUPPORTED	Function is not supported.

Precondition

The component has been initialized. A cryptographic session has been previously setup using the cpaCySymDpInitSession function call.

Postcondition

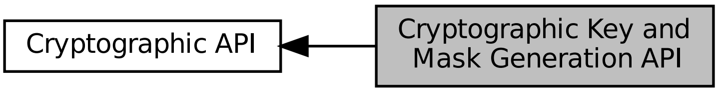
None

See also

cpaCySymDpEnqueueOp, cpaCySymDpEnqueueOpBatch

4.9 Cryptographic Key and Mask Generation API

Collaboration diagram for Cryptographic Key and Mask Generation API:



Data Structures

- struct _CpaCyKeyGenSslOpData
- struct _CpaCyKeyGenHKDFExpandLabel
- struct _CpaCyKeyGenHKDFOpData
- struct _CpaCyKeyGenTlsOpData
- struct _CpaCyKeyGenMgfOpData
- struct _CpaCyKeyGenMgfOpDataExt
- struct _CpaCyKeyGenStats
- struct _CpaCyKeyGenStats64

Macros

- #define CPA_CY_KEY_GEN_SSL_TLS_RANDOM_LEN_IN_BYTES
- #define CPA_CY_HKDF_SUBLABEL_KEY

Typedefs

- typedef enum _CpaCyKeySslOp CpaCyKeySslOp
- typedef struct _CpaCyKeyGenSslOpData CpaCyKeyGenSslOpData
- typedef enum _CpaCyKeyTlsOp CpaCyKeyTlsOp
- typedef enum _CpaCyKeyHKDFOp CpaCyKeyHKDFOp
- typedef enum _CpaCyKeyHKDFCipherSuite CpaCyKeyHKDFCipherSuite
- typedef struct _CpaCyKeyGenHKDFExpandLabel CpaCyKeyGenHKDFExpandLabel
- typedef struct _CpaCyKeyGenHKDFOpData CpaCyKeyGenHKDFOpData
- typedef struct _CpaCyKeyGenTlsOpData CpaCyKeyGenTlsOpData
- typedef struct _CpaCyKeyGenMgfOpData CpaCyKeyGenMgfOpData
- typedef struct _CpaCyKeyGenMgfOpDataExt CpaCyKeyGenMgfOpDataExt
- typedef struct _CpaCyKeyGenStats CPA_DEPRECATED
- typedef struct _CpaCyKeyGenStats64 CpaCyKeyGenStats64

Enumerations

- enum _CpaCyKeySslOp
- enum _CpaCyKeyTlsOp
- enum _CpaCyKeyHKDFOp
- enum _CpaCyKeyHKDFCipherSuite

Functions

- CpaStatus cpaCyKeyGenSsl (const CpaInstanceHandle instanceHandle, const CpaCyGenFlatBufCbFunc pKeyGenCb, void ∗pCallbackTag, const CpaCyKeyGenSslOpData ∗pKeyGenSslOpData, CpaFlatBuffer ∗pGeneratedKeyBuffer)
- CpaStatus cpaCyKeyGenTls (const CpaInstanceHandle instanceHandle, const CpaCyGenFlatBufCbFunc pKeyGenCb, void ∗pCallbackTag, const CpaCyKeyGenTlsOpData ∗pKeyGenTlsOpData, CpaFlatBuffer ∗pGeneratedKeyBuffer)
- CpaStatus cpaCyKeyGenTls2 (const CpaInstanceHandle instanceHandle, const CpaCyGenFlatBufCbFunc pKeyGenCb, void ∗pCallbackTag, const CpaCyKeyGenTlsOpData ∗pKeyGenTlsOpData, CpaCySymHashAlgorithm hashAlgorithm, CpaFlatBuffer ∗pGeneratedKeyBuffer)
- CpaStatus cpaCyKeyGenTls3 (const CpaInstanceHandle instanceHandle, const CpaCyGenFlatBufCbFunc pKeyGenCb, void ∗pCallbackTag, const CpaCyKeyGenHKDFOpData ∗pKeyGenTlsOpData, CpaCyKeyHKDFCipherSuite cipherSuite, CpaFlatBuffer ∗pGeneratedKeyBuffer)
- CpaStatus cpaCyKeyGenMgf (const CpaInstanceHandle instanceHandle, const CpaCyGenFlatBufCbFunc pKeyGenCb, void ∗pCallbackTag, const CpaCyKeyGenMgfOpData ∗pKeyGenMgfOpData, CpaFlatBuffer ∗pGeneratedMaskBuffer)
- CpaStatus cpaCyKeyGenMgfExt (const CpaInstanceHandle instanceHandle, const CpaCyGenFlatBufCbFunc pKeyGenCb, void ∗pCallbackTag, const CpaCyKeyGenMgfOpDataExt ∗pKeyGenMgfOpDataExt, CpaFlatBuffer ∗pGeneratedMaskBuffer)
- CpaStatus CPA_DEPRECATED cpaCyKeyGenQueryStats (const CpaInstanceHandle instanceHandle, struct _CpaCyKeyGenStats ∗pKeyGenStats)
- CpaStatus cpaCyKeyGenQueryStats64 (const CpaInstanceHandle instanceHandle, CpaCyKeyGenStats64 ∗pKeyGenStats)

4.9.1 Detailed Description

File: cpa_cy_key.h

Description:

These functions specify the API for key and mask generation operations.

4.9.2 Macro Definition Documentation

4.9.2.1 CPA_CY_KEY_GEN_SSL_TLS_RANDOM_LEN_IN_BYTES

```
#define CPA_CY_KEY_GEN_SSL_TLS_RANDOM_LEN_IN_BYTES
```

SSL or TLS key generation random number length.

Description:

Defines the permitted SSL or TLS random number length in bytes that may be used with the functions cpaCyKeyGenSsl and cpaCyKeyGenTls. This is the length of the client or server random number values.

Definition at line 47 of file cpa_cy_key.h.

4.9.2.2 CPA_CY_HKDF_SUBLABEL_KEY

```
#define CPA_CY_HKDF_SUBLABEL_KEY
```

File: cpa_cy_key.h

TLS Operation Types

Description:

Bitwise constants for HKDF sublabels

These definitions provide bit settings for sublabels for HKDF-ExpandLabel operations.

key sublabel to generate "key" keying material
iv sublabel to generate "iv" keying material
resumption sublabel to generate "resumption" keying material
finished sublabel to generate "finished" keying material Bit for creation of key material for 'key' sublabel

Definition at line 261 of file cpa_cy_key.h.

4.9.3 Typedef Documentation

4.9.3.1 CpaCyKeySslOp

```
typedef enum _CpaCyKeySslOp CpaCyKeySslOp
```

SSL Operation Types

Description:

Enumeration of the different SSL operations that can be specified in the struct CpaCyKeyGenSslOpData. It identifies the label.

4.9.3.2 CpaCyKeyGenSslOpData

`typedef struct _CpaCyKeyGenSslOpData CpaCyKeyGenSslOpData`

SSL data for key generation functions

Description:

This structure contains data for use in key generation operations for SSL. For specific SSL key generation operations, the structure fields MUST be set as follows:

SSL Master-Secret Derivation:

sslOp = CPA_CY_KEY_SSL_OP_MASTER_SECRET_DERIVE
secret = pre-master secret key
seed = client_random + server_random
userLabel = NULL

SSL Key-Material Derivation:

sslOp = CPA_CY_KEY_SSL_OP_KEY_MATERIAL_DERIVE
secret = master secret key
seed = server_random + client_random
userLabel = NULL

Note that the client/server random order is reversed from that used for master-secret derivation.

Note

Each of the client and server random numbers need to be of length CPA_CY_KEY_GEN_SSL_TLS_RANDOM_LEN_IN_BYTES.

In each of the above descriptions, + indicates concatenation.

The label used is predetermined by the SSL operation in line with the SSL 3.0 specification, and can be overridden by using a user defined operation CPA_CY_KEY_SSL_OP_USER_DEFINED and associated userLabel.

4.9.3.3 CpaCyKeyTlsOp

`typedef enum _CpaCyKeyTlsOp CpaCyKeyTlsOp`

TLS Operation Types

Description:

Enumeration of the different TLS operations that can be specified in the CpaCyKeyGenTlsOpData. It identifies the label.

The functions cpaCyKeyGenTls and cpaCyKeyGenTls2 accelerate the TLS PRF, which is defined as part of RFC2246 (TLS v1.0), RFC4346 (TLS v1.1), and RFC5246 (TLS v1.2). One of the inputs to each of these functions is a label. This enumerated type defines values that correspond to some of the required labels. However, for some of the operations/labels required by these RFCs, no values are specified.

In such cases, a user-defined value must be provided. The client should use the enum value CPA_CY_KEY_TLS_OP_USER_DEFINED, and pass the label using the userLabel field of the CpaCyKeyGenTlsOpData data structure.

4.9.3.4 CpaCyKeyHKDFOp

```
typedef enum _CpaCyKeyHKDFOp CpaCyKeyHKDFOp
```

File: cpa_cy_key.h

TLS Operation Types

Description:

Enumeration of the different TLS operations that can be specified in the CpaCyKeyGenHKDFOpData.

The function cpaCyKeyGenTls3 accelerates the TLS HKDF, which is defined as part of RFC5869 (HKDF) and RFC8446 (TLS v1.3).

This enumerated type defines the support HKDF operations for extraction and expansion of keying material.

4.9.3.5 CpaCyKeyHKDFCipherSuite

```
typedef enum _CpaCyKeyHKDFCipherSuite CpaCyKeyHKDFCipherSuite
```

File: cpa_cy_key.h

TLS Operation Types

Description:

Enumeration of the different cipher suites that may be used in a TLS v1.3 operation. This value is used to infer the sizes of the key and iv sublabel.

The function cpaCyKeyGenTls3 accelerates the TLS HKDF, which is defined as part of RFC5869 (HKDF) and RFC8446 (TLS v1.3).

This enumerated type defines the supported cipher suites in the TLS operation that require HKDF key operations.

4.9.3.6 CpaCyKeyGenHKDFExpandLabel

```
typedef struct _CpaCyKeyGenHKDFExpandLabel CpaCyKeyGenHKDFExpandLabel
```

Maximum number of labels in op structure

File: cpa_cy_key.h

TLS data for key generation functions

Description:

This structure contains data for describing label for the HKDF Extract Label function

Extract Label Function

labelLen = length of the label field
contextLen = length of the context field
sublabelFlag = Mask of sub labels required for this label.
label = label as defined in RFC8446
context = context as defined in RFC8446

4.9.3.7 CpaCyKeyGenHKDFOpData

```
typedef struct _CpaCyKeyGenHKDFOpData CpaCyKeyGenHKDFOpData
```

TLS data for key generation functions

Description:

This structure contains data for all HKDF operations:
HKDF Extract
HKDF Expand
HKDF Expand Label
HKDF Extract and Expand
HKDF Extract and Expand Label

HKDF Map Structure Elements

secret - IKM value for extract operations or PRK for expand or expand operations.
seed - contains the salt for extract operations
info - contains the info data for extract operations
labels - See notes above

4.9.3.8 CpaCyKeyGenTlsOpData

```
typedef struct _CpaCyKeyGenTlsOpData CpaCyKeyGenTlsOpData
```

TLS data for key generation functions

Description:

This structure contains data for use in key generation operations for TLS. For specific TLS key generation operations, the structure fields MUST be set as follows:

TLS Master-Secret Derivation:

tlsOp = CPA_CY_KEY_TLS_OP_MASTER_SECRET_DERIVE
secret = pre-master secret key
seed = client_random + server_random
userLabel = NULL

TLS Key-Material Derivation:

tlsOp = CPA_CY_KEY_TLS_OP_KEY_MATERIAL_DERIVE
secret = master secret key
seed = server_random + client_random
userLabel = NULL

Note that the client/server random order is reversed from that used for Master-Secret Derivation.

TLS Client finished/Server finished tag Derivation:

tlsOp = CPA_CY_KEY_TLS_OP_CLIENT_FINISHED_DERIVE (client)
or CPA_CY_KEY_TLS_OP_SERVER_FINISHED_DERIVE (server)
secret = master secret key
seed = MD5(handshake_messages) + SHA-1(handshake_messages)
userLabel = NULL

Note

Each of the client and server random seeds need to be of length CPA_CY_KEY_GEN_SSL_TLS_RANDOM_LEN_IN_BYTES.

In each of the above descriptions, + indicates concatenation.

The label used is predetermined by the TLS operation in line with the TLS specifications, and can be overridden by using a user defined operation CPA_CY_KEY_TLS_OP_USER_DEFINED and associated userLabel.

4.9.3.9 CpaCyKeyGenMgfOpData

```
typedef struct _CpaCyKeyGenMgfOpData CpaCyKeyGenMgfOpData
```

Key Generation Mask Generation Function (MGF) Data

Description:

This structure contains data relating to Mask Generation Function key generation operations.

Note

The default hash algorithm used by the MGF is SHA-1. If a different hash algorithm is preferred, then see the extended version of this structure, CpaCyKeyGenMgfOpDataExt.

See also

cpaCyKeyGenMgf

4.9.3.10 CpaCyKeyGenMgfOpDataExt

```
typedef struct _CpaCyKeyGenMgfOpDataExt CpaCyKeyGenMgfOpDataExt
```

Extension to the original Key Generation Mask Generation Function (MGF) Data

Description:

This structure is an extension to the original MGF data structure. The extension allows the hash function to be specified.

Note

This structure is separate from the base CpaCyKeyGenMgfOpData structure in order to retain backwards compatibility with the original version of the API.

See also

cpaCyKeyGenMgfExt

4.9.3.11 CPA_DEPRECATED

```
typedef struct _CpaCyKeyGenStats CPA_DEPRECATED
```

Key Generation Statistics.

Deprecated As of v1.3 of the Crypto API, this structure has been deprecated, replaced by CpaCyKeyGenStats64.

Description:

This structure contains statistics on the key and mask generation operations. Statistics are set to zero when the component is initialized, and are collected per instance.

4.9.3.12 CpaCyKeyGenStats64

```
typedef struct _CpaCyKeyGenStats64 CpaCyKeyGenStats64
```

Key Generation Statistics (64-bit version).

Description:

This structure contains the 64-bit version of the statistics on the key and mask generation operations. Statistics are set to zero when the component is initialized, and are collected per instance.

4.9.4 Enumeration Type Documentation

4.9.4.1 _CpaCyKeySslOp

```
enum _CpaCyKeySslOp
```

SSL Operation Types

Description:

Enumeration of the different SSL operations that can be specified in the struct CpaCyKeyGenSslOpData. It identifies the label.

Enumerator

CPA_CY_KEY_SSL_OP_MASTER_SECRET_DERIVE	Derive the master secret
CPA_CY_KEY_SSL_OP_KEY_MATERIAL_DERIVE	Derive the key material
CPA_CY_KEY_SSL_OP_USER_DEFINED	User Defined Operation for custom labels

Definition at line 57 of file cpa_cy_key.h.

4.9.4.2 _CpaCyKeyTlsOp

```
enum _CpaCyKeyTlsOp
```

TLS Operation Types

Description:

Enumeration of the different TLS operations that can be specified in the CpaCyKeyGenTlsOpData. It identifies the label.

The functions cpaCyKeyGenTls and cpaCyKeyGenTls2 accelerate the TLS PRF, which is defined as part of RFC2246 (TLS v1.0), RFC4346 (TLS v1.1), and RFC5246 (TLS v1.2). One of the inputs to each of these functions is a label. This enumerated type defines values that correspond to some of the required labels. However, for some of the operations/labels required by these RFCs, no values are specified.

In such cases, a user-defined value must be provided. The client should use the enum value CPA_CY_KEY_TLS_OP_USER_DEFINED, and pass the label using the userLabel field of the CpaCyKeyGenTlsOpData data structure.

Enumerator

CPA_CY_KEY_TLS_OP_MASTER_SECRET_DE↩ RIVE	Derive the master secret using the TLS PRF. Corresponds to RFC2246/5246 section 8.1, operation "Computing the master secret", label "master secret".
CPA_CY_KEY_TLS_OP_KEY_MATERIAL_DERIVE	Derive the key material using the TLS PRF. Corresponds to RFC2246/5246 section 6.3, operation "Derive the key material", label "key expansion".
CPA_CY_KEY_TLS_OP_CLIENT_FINISHED_DE↩ RIVE	Derive the client finished tag using the TLS PRF. Corresponds to RFC2246/5246 section 7.4.9, operation "Client finished", label "client finished".
CPA_CY_KEY_TLS_OP_SERVER_FINISHED_D↩ ERIVE	Derive the server finished tag using the TLS PRF. Corresponds to RFC2246/5246 section 7.4.9, operation "Server finished", label "server finished".
CPA_CY_KEY_TLS_OP_USER_DEFINED	User Defined Operation for custom labels.

Definition at line 150 of file cpa_cy_key.h.

4.9.4.3 _CpaCyKeyHKDFOp

```
enum _CpaCyKeyHKDFOp
```

File: cpa_cy_key.h

TLS Operation Types

Description:

Enumeration of the different TLS operations that can be specified in the CpaCyKeyGenHKDFOpData.

The function cpaCyKeyGenTls3 accelerates the TLS HKDF, which is defined as part of RFC5869 (HKDF) and RFC8446 (TLS v1.3).

This enumerated type defines the support HKDF operations for extraction and expansion of keying material.

Enumerator

CPA_CY_HKDF_KEY_EXTRACT	HKDF Extract operation Corresponds to RFC5869 section 2.2, step 1 "Extract"
CPA_CY_HKDF_KEY_EXPAND	HKDF Expand operation Corresponds to RFC5869 section 2.3, step 2 "Expand"
CPA_CY_HKDF_KEY_EXTRACT_EXPAND	HKDF operation This performs HKDF_EXTRACT and HKDF_EXPAND in a single API invocation.
CPA_CY_HKDF_KEY_EXPAND_LABEL	HKDF Expand label operation for TLS 1.3 Corresponds to RFC8446 section 7.1 Key Schedule definition for HKDF-Expand-Label, which refers to HKDF-Expand defined in RFC5869.
CPA_CY_HKDF_KEY_EXTRACT_EXPAND_LABEL	HKDF Extract plus Expand label operation for TLS 1.3 Corresponds to RFC5869 section 2.2, step 1 "Extract" followed by RFC8446 section 7.1 Key Schedule definition for HKDF-Expand-Label, which refers to HKDF-Expand defined in RFC5869.

Definition at line 191 of file cpa_cy_key.h.

4.9.4.4 _CpaCyKeyHKDFCipherSuite

```
enum _CpaCyKeyHKDFCipherSuite
```

File: cpa_cy_key.h

TLS Operation Types

Description:

Enumeration of the different cipher suites that may be used in a TLS v1.3 operation. This value is used to infer the sizes of the key and iv sublabel.

The function cpaCyKeyGenTls3 accelerates the TLS HKDF, which is defined as part of RFC5869 (HKDF) and RFC8446 (TLS v1.3).

This enumerated type defines the supported cipher suites in the TLS operation that require HKDF key operations.

Definition at line 233 of file cpa_cy_key.h.

4.9.5 Function Documentation

4.9.5.1 cpaCyKeyGenSsl()

```
CpaStatus cpaCyKeyGenSsl (
            const CpaInstanceHandle instanceHandle,
            const CpaCyGenFlatBufCbFunc pKeyGenCb,
            void * pCallbackTag,
            const CpaCyKeyGenSslOpData * pKeyGenSslOpData,
            CpaFlatBuffer * pGeneratedKeyBuffer )
```

SSL Key Generation Function.

Description:

This function is used for SSL key generation. It implements the key generation function defined in section 6.2.2 of the SSL 3.0 specification as described in http://www.mozilla.org/projects/security/pki/nss/ssl/draft302.txt.

The input seed is taken as a flat buffer and the generated key is returned to caller in a flat destination data buffer.

Context:

When called as an asynchronous function it cannot sleep. It can be executed in a context that does not permit sleeping. When called as a synchronous function it may sleep. It MUST NOT be executed in a context that DOES NOT permit sleeping.

Assumptions:

None

Side-Effects:

None

Blocking:

Yes when configured to operate in synchronous mode.

Reentrant:

No

Thread-safe:

Yes

Parameters

in	*instanceHandle*	Instance handle.
in	*pKeyGenCb*	Pointer to callback function to be invoked when the operation is complete. If this is set to a NULL value the function will operate synchronously.
in	*pCallbackTag*	Opaque User Data for this specific call. Will be returned unchanged in the callback.
in	*pKeyGenSslOpData*	Structure containing all the data needed to perform the SSL key generation operation. The client code allocates the memory for this structure. This component takes ownership of the memory until it is returned in the callback.
out	*pGeneratedKeyBuffer*	Caller MUST allocate a sufficient buffer to hold the key generation output. The data pointer SHOULD be aligned on an 8-byte boundary. The length field passed in represents the size of the buffer in bytes. The value that is returned is the size of the result key in bytes. On invocation the callback function will contain this parameter in the pOut parameter.

Return values

CPA_STATUS_SUCCESS	Function executed successfully.
CPA_STATUS_FAIL	Function failed.
CPA_STATUS_RETRY	Resubmit the request.
CPA_STATUS_INVALID_PARAM	Invalid parameter passed in.
CPA_STATUS_RESOURCE	Error related to system resources.
CPA_STATUS_RESTARTING	API implementation is restarting. Resubmit the request.

Precondition

The component has been initialized via cpaCyStartInstance function.

Postcondition

None

See also

CpaCyKeyGenSslOpData, CpaCyGenFlatBufCbFunc

4.9.5.2 cpaCyKeyGenTls()

```
CpaStatus cpaCyKeyGenTls (
            const CpaInstanceHandle instanceHandle,
            const CpaCyGenFlatBufCbFunc pKeyGenCb,
            void * pCallbackTag,
            const CpaCyKeyGenTlsOpData * pKeyGenTlsOpData,
            CpaFlatBuffer * pGeneratedKeyBuffer )
```

TLS Key Generation Function.

Description:

This function is used for TLS key generation. It implements the TLS PRF (Pseudo Random Function) as defined by RFC2246 (TLS v1.0) and RFC4346 (TLS v1.1).

The input seed is taken as a flat buffer and the generated key is returned to caller in a flat destination data buffer.

Context:

When called as an asynchronous function it cannot sleep. It can be executed in a context that does not permit sleeping. When called as a synchronous function it may sleep. It MUST NOT be executed in a context that DOES NOT permit sleeping.

Assumptions:

None

Side-Effects:

None

Blocking:

Yes when configured to operate in synchronous mode.

Reentrant:

No

Thread-safe:

Yes

Parameters

in	*instanceHandle*	Instance handle.
in	*pKeyGenCb*	Pointer to callback function to be invoked when the operation is complete. If this is set to a NULL value the function will operate synchronously.
in	*pCallbackTag*	Opaque User Data for this specific call. Will be returned unchanged in the callback.
in	*pKeyGenTlsOpData*	Structure containing all the data needed to perform the TLS key generation operation. The client code allocates the memory for this structure. This component takes ownership of the memory until it is returned in the callback.
out	*pGeneratedKeyBuffer*	Caller MUST allocate a sufficient buffer to hold the key generation output. The data pointer SHOULD be aligned on an 8-byte boundary. The length field passed in represents the size of the buffer in bytes. The value that is returned is the size of the result key in bytes. On invocation the callback function will contain this parameter in the pOut parameter.

Return values

CPA_STATUS_SUCCESS	Function executed successfully.
CPA_STATUS_FAIL	Function failed.
CPA_STATUS_RETRY	Resubmit the request.
CPA_STATUS_INVALID_PARAM	Invalid parameter passed in.
CPA_STATUS_RESOURCE	Error related to system resources.
CPA_STATUS_RESTARTING	API implementation is restarting. Resubmit the request.

Precondition

The component has been initialized via cpaCyStartInstance function.

Postcondition

None

See also

CpaCyKeyGenTlsOpData, CpaCyGenFlatBufCbFunc

4.9.5.3 cpaCyKeyGenTls2()

```
CpaStatus cpaCyKeyGenTls2 (
              const CpaInstanceHandle instanceHandle,
              const CpaCyGenFlatBufCbFunc pKeyGenCb,
              void * pCallbackTag,
              const CpaCyKeyGenTlsOpData * pKeyGenTlsOpData,
              CpaCySymHashAlgorithm hashAlgorithm,
              CpaFlatBuffer * pGeneratedKeyBuffer )
```

TLS Key Generation Function version 2.

Description:

This function is used for TLS key generation. It implements the TLS PRF (Pseudo Random Function) as defined by RFC5246 (TLS v1.2).

The input seed is taken as a flat buffer and the generated key is returned to caller in a flat destination data buffer.

Context:

When called as an asynchronous function it cannot sleep. It can be executed in a context that does not permit sleeping. When called as a synchronous function it may sleep. It MUST NOT be executed in a context that DOES NOT permit sleeping.

Assumptions:

None

Side-Effects:

None

Blocking:

Yes when configured to operate in synchronous mode.

Reentrant:

No

Thread-safe:

Yes

Parameters

in	*instanceHandle*	Instance handle.
in	*pKeyGenCb*	Pointer to callback function to be invoked when the operation is complete. If this is set to a NULL value the function will operate synchronously.
in	*pCallbackTag*	Opaque User Data for this specific call. Will be returned unchanged in the callback.
in	*pKeyGenTlsOpData*	Structure containing all the data needed to perform the TLS key generation operation. The client code allocates the memory for this structure. This component takes ownership of the memory until it is returned in the callback.
in	*hashAlgorithm*	Specifies the hash algorithm to use. According to RFC5246, this should be "SHA-256 or a stronger standard hash function."
out	*pGeneratedKeyBuffer*	Caller MUST allocate a sufficient buffer to hold the key generation output. The data pointer SHOULD be aligned on an 8-byte boundary. The length field passed in represents the size of the buffer in bytes. The value that is returned is the size of the result key in bytes. On invocation the callback function will contain this parameter in the pOut parameter.

Return values

CPA_STATUS_SUCCESS	Function executed successfully.
CPA_STATUS_FAIL	Function failed.
CPA_STATUS_RETRY	Resubmit the request.
CPA_STATUS_INVALID_PARAM	Invalid parameter passed in.
CPA_STATUS_RESOURCE	Error related to system resources.
CPA_STATUS_RESTARTING	API implementation is restarting. Resubmit the request.

Precondition

The component has been initialized via cpaCyStartInstance function.

Postcondition

None

See also

CpaCyKeyGenTlsOpData, CpaCyGenFlatBufCbFunc

4.9.5.4 cpaCyKeyGenTls3()

```
CpaStatus cpaCyKeyGenTls3 (
            const CpaInstanceHandle instanceHandle,
            const CpaCyGenFlatBufCbFunc pKeyGenCb,
            void * pCallbackTag,
            const CpaCyKeyGenHKDFOpData * pKeyGenTlsOpData,
            CpaCyKeyHKDFCipherSuite cipherSuite,
            CpaFlatBuffer * pGeneratedKeyBuffer )
```

TLS Key Generation Function version 3.

Description:

This function is used for TLS key generation. It implements the TLS HKDF (HMAC Key Derivation Function) as defined by RFC5689 (HKDF) and RFC8446 (TLS 1.3).

The input seed is taken as a flat buffer and the generated key is returned to caller in a flat destination data buffer.

Context:

When called as an asynchronous function it cannot sleep. It can be executed in a context that does not permit sleeping. When called as a synchronous function it may sleep. It MUST NOT be executed in a context that DOES NOT permit sleeping.

Assumptions:

None

Side-Effects:

None

Blocking:

Yes when configured to operate in synchronous mode.

Reentrant:

No

Thread-safe:

Yes

Parameters

`in`	*instanceHandle*	Instance handle.
`in`	*pKeyGenCb*	Pointer to callback function to be invoked when the operation is complete. If this is set to a NULL value the function will operate synchronously.
`in`	*pCallbackTag*	Opaque User Data for this specific call. Will be returned unchanged in the callback.
`in`	*pKeyGenTlsOpData*	Structure containing all the data needed to perform the TLS key generation operation. The client code allocates the memory for this structure. This component takes ownership of the memory until it is returned in the callback. The memory must be pinned and contiguous, suitable for DMA operations.
`in`	*hashAlgorithm*	Specifies the hash algorithm to use. According to RFC5246, this should be "SHA-256 or a stronger standard hash function."
`out`	*pGeneratedKeyBuffer*	Caller MUST allocate a sufficient buffer to hold the key generation output. The data pointer SHOULD be aligned on an 8-byte boundary. The length field passed in represents the size of the buffer in bytes. The value that is returned is the size of the result key in bytes. On invocation the callback function will contain this parameter in the pOut parameter.

Return values

CPA_STATUS_SUCCESS	Function executed successfully.
CPA_STATUS_FAIL	Function failed.
CPA_STATUS_RETRY	Resubmit the request.
CPA_STATUS_INVALID_PARAM	Invalid parameter passed in.
CPA_STATUS_RESOURCE	Error related to system resources.
CPA_STATUS_RESTARTING	API implementation is restarting. Resubmit the request.

Precondition

The component has been initialized via cpaCyStartInstance function.

Postcondition

None

See also

CpaCyGenFlatBufCbFunc CpaCyKeyGenHKDFOpData

4.9.5.5 cpaCyKeyGenMgf()

```
CpaStatus cpaCyKeyGenMgf (
            const CpaInstanceHandle instanceHandle,
            const CpaCyGenFlatBufCbFunc pKeyGenCb,
            void * pCallbackTag,
            const CpaCyKeyGenMgfOpData * pKeyGenMgfOpData,
            CpaFlatBuffer * pGeneratedMaskBuffer )
```

Mask Generation Function.

Description:

This function implements the mask generation function MGF1 as defined by PKCS#1 v2.1, and RFC3447. The input seed is taken as a flat buffer and the generated mask is returned to caller in a flat destination data buffer.

Note

The default hash algorithm used by the MGF is SHA-1. If a different hash algorithm is preferred, then see the "extended" version of this function, cpaCyKeyGenMgfExt.

Context:

When called as an asynchronous function it cannot sleep. It can be executed in a context that does not permit sleeping. When called as a synchronous function it may sleep. It MUST NOT be executed in a context that DOES NOT permit sleeping.

Assumptions:

None

Side-Effects:

None

Blocking:

Yes when configured to operate in synchronous mode.

Reentrant:

No

Thread-safe:

Yes

Parameters

`in`	*instanceHandle*	Instance handle.
`in`	*pKeyGenCb*	Pointer to callback function to be invoked when the operation is complete. If this is set to a NULL value the function will operate synchronously.
`in`	*pCallbackTag*	Opaque User Data for this specific call. Will be returned unchanged in the callback.
`in`	*pKeyGenMgfOpData*	Structure containing all the data needed to perform the MGF key generation operation. The client code allocates the memory for this structure. This component takes ownership of the memory until it is returned in the callback.
`out`	*pGeneratedMaskBuffer*	Caller MUST allocate a sufficient buffer to hold the generated mask. The data pointer SHOULD be aligned on an 8-byte boundary. The length field passed in represents the size of the buffer in bytes. The value that is returned is the size of the generated mask in bytes. On invocation the callback function will contain this parameter in the pOut parameter.

Return values

CPA_STATUS_SUCCESS	Function executed successfully.
CPA_STATUS_FAIL	Function failed.
CPA_STATUS_RETRY	Resubmit the request.
CPA_STATUS_INVALID_PARAM	Invalid parameter passed in.
CPA_STATUS_RESOURCE	Error related to system resources.
CPA_STATUS_RESTARTING	API implementation is restarting. Resubmit the request.

Precondition

The component has been initialized via cpaCyStartInstance function.

Postcondition

None

See also

CpaCyKeyGenMgfOpData, CpaCyGenFlatBufCbFunc

4.9.5.6 cpaCyKeyGenMgfExt()

```
CpaStatus cpaCyKeyGenMgfExt (
            const CpaInstanceHandle instanceHandle,
            const CpaCyGenFlatBufCbFunc pKeyGenCb,
            void * pCallbackTag,
            const CpaCyKeyGenMgfOpDataExt * pKeyGenMgfOpDataExt,
            CpaFlatBuffer * pGeneratedMaskBuffer )
```

Extended Mask Generation Function.

Description:

This function is used for mask generation. It differs from the "base" version of the function (cpaCyKeyGenMgf) in that it allows the hash function used by the Mask Generation Function to be specified.

Context:

When called as an asynchronous function it cannot sleep. It can be executed in a context that does not permit sleeping. When called as a synchronous function it may sleep. It MUST NOT be executed in a context that DOES NOT permit sleeping.

Assumptions:

None

Side-Effects:

None

Blocking:

Yes when configured to operate in synchronous mode.

Reentrant:

No

Thread-safe:

Yes

Parameters

in	*instanceHandle*	Instance handle.
in	*pKeyGenCb*	Pointer to callback function to be invoked when the operation is complete. If this is set to a NULL value the function will operate synchronously.
in	*pCallbackTag*	Opaque User Data for this specific call. Will be returned unchanged in the callback.
in	*pKeyGenMgfOpDataExt*	Structure containing all the data needed to perform the extended MGF key generation operation. The client code allocates the memory for this structure. This component takes ownership of the memory until it is returned in the callback.
out	*pGeneratedMaskBuffer*	Caller MUST allocate a sufficient buffer to hold the generated mask. The data pointer SHOULD be aligned on an 8-byte boundary. The length field passed in represents the size of the buffer in bytes. The value that is returned is the size of the generated mask in bytes. On invocation the callback function will contain this parameter in the pOut parameter.

Return values

CPA_STATUS_SUCCESS	Function executed successfully.
CPA_STATUS_FAIL	Function failed.
CPA_STATUS_RETRY	Resubmit the request.
CPA_STATUS_INVALID_PARAM	Invalid parameter passed in.
CPA_STATUS_RESOURCE	Error related to system resources.
CPA_STATUS_RESTARTING	API implementation is restarting. Resubmit the request.

Precondition

The component has been initialized via cpaCyStartInstance function.

Postcondition

None

Note

This function is only used to generate a mask keys from seed material.

See also

CpaCyKeyGenMgfOpData, CpaCyGenFlatBufCbFunc

4.9.5.7 cpaCyKeyGenQueryStats()

```
CpaStatus CPA_DEPRECATED cpaCyKeyGenQueryStats (
            const CpaInstanceHandle instanceHandle,
            struct _CpaCyKeyGenStats * pKeyGenStats )
```

Queries the Key and Mask generation statistics specific to an instance.

Deprecated As of v1.3 of the Crypto API, this function has been deprecated, replaced by cpaCyKeyGenQueryStats64().

Description:

This function will query a specific instance for key and mask generation statistics. The user MUST allocate the CpaCyKeyGenStats structure and pass the reference to that into this function call. This function will write the statistic results into the passed in CpaCyKeyGenStats structure.

Note: statistics returned by this function do not interrupt current data processing and as such can be slightly out of sync with operations that are in progress during the statistics retrieval process.

Context:

This is a synchronous function and it can sleep. It MUST NOT be executed in a context that DOES NOT permit sleeping.

Assumptions:

None

Side-Effects:

None

Blocking:

This function is synchronous and blocking.

Reentrant:

No

Thread-safe:

Yes

Parameters

in	*instanceHandle*	Instance handle.
out	*pKeyGenStats*	Pointer to memory into which the statistics will be written.

Return values

CPA_STATUS_SUCCESS	Function executed successfully.
CPA_STATUS_FAIL	Function failed.
CPA_STATUS_INVALID_PARAM	Invalid parameter passed in.
CPA_STATUS_RESOURCE	Error related to system resources.
CPA_STATUS_RESTARTING	API implementation is restarting. Resubmit the request.

Precondition

Component has been initialized.

Postcondition

None

Note

This function operates in a synchronous manner and no asynchronous callback will be generated.

See also

CpaCyKeyGenStats

4.9.5.8 cpaCyKeyGenQueryStats64()

```
CpaStatus cpaCyKeyGenQueryStats64 (
            const CpaInstanceHandle instanceHandle,
            CpaCyKeyGenStats64 * pKeyGenStats )
```

Queries the Key and Mask generation statistics (64-bit version) specific to an instance.

Description:

This function will query a specific instance for key and mask generation statistics. The user MUST allocate the CpaCyKeyGenStats64 structure and pass the reference to that into this function call. This function will write the statistic results into the passed in CpaCyKeyGenStats64 structure.

Note: statistics returned by this function do not interrupt current data processing and as such can be slightly out of sync with operations that are in progress during the statistics retrieval process.

Context:

This is a synchronous function and it can sleep. It MUST NOT be executed in a context that DOES NOT permit sleeping.

Assumptions:

None

Side-Effects:

None

Blocking:

This function is synchronous and blocking.

Reentrant:

No

Thread-safe:

Yes

Parameters

in	*instanceHandle*	Instance handle.
out	*pKeyGenStats*	Pointer to memory into which the statistics will be written.

Return values

CPA_STATUS_SUCCESS	Function executed successfully.
CPA_STATUS_FAIL	Function failed.
CPA_STATUS_INVALID_PARAM	Invalid parameter passed in.
CPA_STATUS_RESOURCE	Error related to system resources.
CPA_STATUS_RESTARTING	API implementation is restarting. Resubmit the request.

Precondition

Component has been initialized.

Postcondition

None

Note

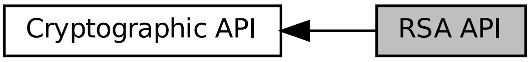
This function operates in a synchronous manner and no asynchronous callback will be generated.

See also

CpaCyKeyGenStats64

4.10 RSA API

Collaboration diagram for RSA API:



Data Structures

- struct _CpaCyRsaPublicKey
- struct _CpaCyRsaPrivateKeyRep1
- struct _CpaCyRsaPrivateKeyRep2
- struct _CpaCyRsaPrivateKey
- struct _CpaCyRsaKeyGenOpData
- struct _CpaCyRsaEncryptOpData
- struct _CpaCyRsaDecryptOpData
- struct _CpaCyRsaStats
- struct _CpaCyRsaStats64

Typedefs

- typedef enum _CpaCyRsaVersion CpaCyRsaVersion
- typedef struct _CpaCyRsaPublicKey CpaCyRsaPublicKey
- typedef struct _CpaCyRsaPrivateKeyRep1 CpaCyRsaPrivateKeyRep1
- typedef struct _CpaCyRsaPrivateKeyRep2 CpaCyRsaPrivateKeyRep2
- typedef enum _CpaCyRsaPrivateKeyRepType CpaCyRsaPrivateKeyRepType
- typedef struct _CpaCyRsaPrivateKey CpaCyRsaPrivateKey
- typedef struct _CpaCyRsaKeyGenOpData CpaCyRsaKeyGenOpData
- typedef struct _CpaCyRsaEncryptOpData CpaCyRsaEncryptOpData
- typedef struct _CpaCyRsaDecryptOpData CpaCyRsaDecryptOpData
- typedef struct _CpaCyRsaStats CPA_DEPRECATED
- typedef struct _CpaCyRsaStats64 CpaCyRsaStats64
- typedef void(∗ CpaCyRsaKeyGenCbFunc) (void ∗pCallbackTag, CpaStatus status, void ∗pKeyGenOpData, CpaCyRsaPrivateKey ∗pPrivateKey, CpaCyRsaPublicKey ∗pPublicKey)

Enumerations

- enum _CpaCyRsaVersion
- enum _CpaCyRsaPrivateKeyRepType

Functions

- CpaStatus cpaCyRsaGenKey (const CpaInstanceHandle instanceHandle, const CpaCyRsaKeyGenCbFunc pRsaKeyGenCb, void ∗pCallbackTag, const CpaCyRsaKeyGenOpData ∗pKeyGenOpData, CpaCyRsaPrivateKey ∗pPrivateKey, CpaCyRsaPublicKey ∗pPublicKey)
- CpaStatus cpaCyRsaEncrypt (const CpaInstanceHandle instanceHandle, const CpaCyGenFlatBufCbFunc pRsaEncryptCb, void ∗pCallbackTag, const CpaCyRsaEncryptOpData ∗pEncryptOpData, CpaFlatBuffer ∗pOutputData)
- CpaStatus cpaCyRsaDecrypt (const CpaInstanceHandle instanceHandle, const CpaCyGenFlatBufCbFunc pRsaDecryptCb, void ∗pCallbackTag, const CpaCyRsaDecryptOpData ∗pDecryptOpData, CpaFlatBuffer ∗pOutputData)
- CpaStatus CPA_DEPRECATED cpaCyRsaQueryStats (const CpaInstanceHandle instanceHandle, struct _CpaCyRsaStats ∗pRsaStats)
- CpaStatus cpaCyRsaQueryStats64 (const CpaInstanceHandle instanceHandle, CpaCyRsaStats64 ∗pRsaStats)

4.10.1 Detailed Description

File: cpa_cy_rsa.h

Description:

These functions specify the API for Public Key Encryption (Cryptography) RSA operations. The PKCS #1 V2.1 specification is supported, however the support is limited to "two-prime" mode. RSA multi-prime is not supported.

Note

These functions implement RSA cryptographic primitives. RSA padding schemes are not implemented. For padding schemes that require the mgf function see Cryptographic Key and Mask Generation API.

Large numbers are represented on the QuickAssist API as described in the Large Number API (Cryptographic Large Number API).

4.10.2 Typedef Documentation

4.10.2.1 CpaCyRsaVersion

```
typedef enum _CpaCyRsaVersion CpaCyRsaVersion
```

RSA Version.

Description:

This enumeration lists the version identifier for the PKCS #1 V2.1 standard.

Note

Multi-prime (more than two primes) is not supported.

4.10.2.2 CpaCyRsaPublicKey

```
typedef struct _CpaCyRsaPublicKey CpaCyRsaPublicKey
```

RSA Public Key Structure.

Description:

This structure contains the two components which comprise the RSA public key as defined in the PKCS #1 V2.1 standard. All values in this structure are required to be in Most Significant Byte first order, e.g. modulusN.pData[0] = MSB.

4.10.2.3 CpaCyRsaPrivateKeyRep1

```
typedef struct _CpaCyRsaPrivateKeyRep1 CpaCyRsaPrivateKeyRep1
```

RSA Private Key Structure For Representation 1.

Description:

This structure contains the first representation that can be used for describing the RSA private key, represented by the tuple of the modulus (n) and the private exponent (d). All values in this structure are required to be in Most Significant Byte first order, e.g. modulusN.pData[0] = MSB.

4.10.2.4 CpaCyRsaPrivateKeyRep2

```
typedef struct _CpaCyRsaPrivateKeyRep2 CpaCyRsaPrivateKeyRep2
```

RSA Private Key Structure For Representation 2.

Description:

This structure contains the second representation that can be used for describing the RSA private key. The quintuple of p, q, dP, dQ, and qInv (explained below and in the spec) are required for the second representation. The optional sequence of triplets are not included. All values in this structure are required to be in Most Significant Byte first order, e.g. prime1P.pData[0] = MSB.

4.10.2.5 CpaCyRsaPrivateKeyRepType

```
typedef enum _CpaCyRsaPrivateKeyRepType CpaCyRsaPrivateKeyRepType
```

RSA private key representation type.

Description:

This enumeration lists which PKCS V2.1 representation of the private key is being used.

4.10.2.6 CpaCyRsaPrivateKey

```
typedef struct _CpaCyRsaPrivateKey CpaCyRsaPrivateKey
```

RSA Private Key Structure.

Description:

This structure contains the two representations that can be used for describing the RSA private key. The privateKeyRepType will be used to identify which representation is to be used. Typically, using the second representation results in faster decryption operations.

4.10.2.7 CpaCyRsaKeyGenOpData

```
typedef struct _CpaCyRsaKeyGenOpData CpaCyRsaKeyGenOpData
```

RSA Key Generation Data.

Description:

This structure lists the different items that are required in the cpaCyRsaGenKey function. The client MUST allocate the memory for this structure. When the structure is passed into the function, ownership of the memory passes to the function. Ownership of the memory returns to the client when this structure is returned in the CpaCyRsaKeyGenCbFunc callback function.

Note

If the client modifies or frees the memory referenced in this structure after it has been submitted to the cpaCyRsaGenKey function, and before it has been returned in the callback, undefined behavior will result. All values in this structure are required to be in Most Significant Byte first order, e.g. prime1P.pData[0] = MSB.

The following limitations on the permutations of the supported bit lengths of p, q and n (written as {p, q, n}) apply:

- {256, 256, 512} or
- {512, 512, 1024} or
- {768, 768, 1536} or
- {1024, 1024, 2048} or
- {1536, 1536, 3072} or
- {2048, 2048, 4096}.

4.10.2.8 CpaCyRsaEncryptOpData

```
typedef struct _CpaCyRsaEncryptOpData CpaCyRsaEncryptOpData
```

RSA Encryption Primitive Operation Data

Description:

This structure lists the different items that are required in the cpaCyRsaEncrypt function. As the RSA encryption primitive and verification primitive operations are mathematically identical this structure may also be used to perform an RSA verification primitive operation. When performing an RSA encryption primitive operation, the input data is the message and the output data is the cipher text. When performing an RSA verification primitive operation, the input data is the signature and the output data is the message. The client MUST allocate the memory for this structure. When the structure is passed into the function, ownership of the memory passes to the function. Ownership of the memory returns to the client when this structure is returned in the CpaCyRsaEncryptCbFunc callback function.

Note

If the client modifies or frees the memory referenced in this structure after it has been submitted to the cpaCyRsaEncrypt function, and before it has been returned in the callback, undefined behavior will result. All values in this structure are required to be in Most Significant Byte first order, e.g. inputData.pData[0] = MSB.

4.10.2.9 CpaCyRsaDecryptOpData

```
typedef struct _CpaCyRsaDecryptOpData CpaCyRsaDecryptOpData
```

RSA Decryption Primitive Operation Data

Description:

This structure lists the different items that are required in the cpaCyRsaDecrypt function. As the RSA decryption primitive and signature primitive operations are mathematically identical this structure may also be used to perform an RSA signature primitive operation. When performing an RSA decryption primitive operation, the input data is the cipher text and the output data is the message text. When performing an RSA signature primitive operation, the input data is the message and the output data is the signature. The client MUST allocate the memory for this structure. When the structure is passed into the function, ownership of the memory passes to he function. Ownership of the memory returns to the client when this structure is returned in the CpaCyRsaDecryptCbFunc callback function.

Note

If the client modifies or frees the memory referenced in this structure after it has been submitted to the cpaCyRsaDecrypt function, and before it has been returned in the callback, undefined behavior will result. All values in this structure are required to be in Most Significant Byte first order, e.g. inputData.pData[0] = MSB.

4.10.2.10 CPA_DEPRECATED

```
typedef struct _CpaCyRsaStats CPA_DEPRECATED
```

RSA Statistics.

Deprecated As of v1.3 of the Crypto API, this structure has been deprecated, replaced by CpaCyRsaStats64.

Description:

This structure contains statistics on the RSA operations. Statistics are set to zero when the component is initialized, and are collected per instance.

4.10.2.11 CpaCyRsaStats64

```
typedef struct _CpaCyRsaStats64 CpaCyRsaStats64
```

RSA Statistics (64-bit version).

Description:

This structure contains 64-bit version of the statistics on the RSA operations. Statistics are set to zero when the component is initialized, and are collected per instance.

4.10.2.12 CpaCyRsaKeyGenCbFunc

```
typedef void(* CpaCyRsaKeyGenCbFunc) (void *pCallbackTag, CpaStatus status, void
*pKeyGenOpData, CpaCyRsaPrivateKey *pPrivateKey, CpaCyRsaPublicKey *pPublicKey)
```

Definition of the RSA key generation callback function.

Description:

This is the prototype for the RSA key generation callback function. The callback function pointer is passed in as a parameter to the cpaCyRsaGenKey function. It will be invoked once the request has completed.

Context:

This callback function can be executed in a context that DOES NOT permit sleeping to occur.

Assumptions:

None

Side-Effects:

None

Reentrant:

No

Thread-safe:

Yes

Parameters

in	*pCallbackTag*	Opaque value provided by user while making individual function calls.
in	*status*	Status of the operation. Valid values are CPA_STATUS_SUCCESS, CPA_STATUS_FAIL and CPA_STATUS_UNSUPPORTED.
in	*pKeyGenOpData*	Structure with output params for callback.
in	*pPrivateKey*	Structure which contains pointers to the memory into which the generated private key will be written.
in	*pPublicKey*	Structure which contains pointers to the memory into which the generated public key will be written. The pointer to the public exponent (e) that is returned in this structure is equal to the input public exponent.

Return values

None	

Precondition

Component has been initialized.

Postcondition

None

Note

None

See also

CpaCyRsaPrivateKey, CpaCyRsaPublicKey, cpaCyRsaGenKey()

Definition at line 505 of file cpa_cy_rsa.h.

4.10.3 Enumeration Type Documentation

4.10.3.1 _CpaCyRsaVersion

`enum _CpaCyRsaVersion`

RSA Version.

Description:

This enumeration lists the version identifier for the PKCS #1 V2.1 standard.

Note

Multi-prime (more than two primes) is not supported.

Enumerator

CPA_CY_RSA_VERSION_TWO_PRIME	The version supported is "two-prime".

Definition at line 56 of file cpa_cy_rsa.h.

4.10.3.2 _CpaCyRsaPrivateKeyRepType

```
enum _CpaCyRsaPrivateKeyRepType
```

RSA private key representation type.

Description:

This enumeration lists which PKCS V2.1 representation of the private key is being used.

Enumerator

CPA_CY_RSA_PRIVATE_KEY_REP_TYPE↵ _1	The first representation of the RSA private key.
CPA_CY_RSA_PRIVATE_KEY_REP_TYPE↵ _2	The second representation of the RSA private key.

Definition at line 162 of file cpa_cy_rsa.h.

4.10.4 Function Documentation

4.10.4.1 cpaCyRsaGenKey()

```
CpaStatus cpaCyRsaGenKey (
            const CpaInstanceHandle instanceHandle,
            const CpaCyRsaKeyGenCbFunc pRsaKeyGenCb,
            void * pCallbackTag,
            const CpaCyRsaKeyGenOpData * pKeyGenOpData,
            CpaCyRsaPrivateKey * pPrivateKey,
            CpaCyRsaPublicKey * pPublicKey )
```

Generate RSA keys.

Description:

This function will generate private and public keys for RSA as specified in the PKCS #1 V2.1 standard. Both representation types of the private key may be generated.

Context:

When called as an asynchronous function it cannot sleep. It can be executed in a context that does not permit sleeping. When called as a synchronous function it may sleep. It MUST NOT be executed in a context that DOES NOT permit sleeping.

Assumptions:

None

Side-Effects:

None

Blocking:

Yes when configured to operate in synchronous mode.

Reentrant:

No

Thread-safe:

Yes

Parameters

in	*instanceHandle*	Instance handle.
in	*pRsaKeyGenCb*	Pointer to the callback function to be invoked when the operation is complete. If this is set to a NULL value the function will operate synchronously.
in	*pCallbackTag*	Opaque User Data for this specific call. Will be returned unchanged in the callback.
in	*pKeyGenOpData*	Structure containing all the data needed to perform the RSA key generation operation. The client code allocates the memory for this structure. This component takes ownership of the memory until it is returned in the callback.
out	*pPrivateKey*	Structure which contains pointers to the memory into which the generated private key will be written. The client MUST allocate memory for this structure, and for the pointers within it, recursively; on return, these will be populated.
out	*pPublicKey*	Structure which contains pointers to the memory into which the generated public key will be written. The memory for this structure and for the modulusN parameter MUST be allocated by the client, and will be populated on return from the call. The field publicExponentE is not modified or touched in any way; it is the responsibility of the client to set this to the same value as the corresponding parameter on the CpaCyRsaKeyGenOpData structure before using the key for encryption.

Return values

CPA_STATUS_SUCCESS	Function executed successfully.
CPA_STATUS_FAIL	Function failed.
CPA_STATUS_RETRY	Resubmit the request.

Return values

CPA_STATUS_INVALID_PARAM	Invalid parameter passed in.
CPA_STATUS_RESOURCE	Error related to system resources.
CPA_STATUS_RESTARTING	API implementation is restarting. Resubmit the request.
CPA_STATUS_UNSUPPORTED	Function is not supported.

Precondition

The component has been initialized via cpaCyStartInstance function.

Postcondition

None

Note

When pRsaKeyGenCb is non-NULL, an asynchronous callback of type is generated in response to this function call. Any errors generated during processing are reported as part of the callback status code. For optimal performance, data pointers SHOULD be 8-byte aligned.

See also

CpaCyRsaKeyGenOpData, CpaCyRsaKeyGenCbFunc, cpaCyRsaEncrypt(), cpaCyRsaDecrypt()

4.10.4.2 cpaCyRsaEncrypt()

```
CpaStatus cpaCyRsaEncrypt (
            const CpaInstanceHandle instanceHandle,
            const CpaCyGenFlatBufCbFunc pRsaEncryptCb,
            void * pCallbackTag,
            const CpaCyRsaEncryptOpData * pEncryptOpData,
            CpaFlatBuffer * pOutputData )
```

Perform the RSA encrypt (or verify) primitive operation on the input data.

Description:

This function will perform an RSA encryption primitive operation on the input data using the specified RSA public key. As the RSA encryption primitive and verification primitive operations are mathematically identical this function may also be used to perform an RSA verification primitive operation.

Context:

When called as an asynchronous function it cannot sleep. It can be executed in a context that does not permit sleeping. When called as a synchronous function it may sleep. It MUST NOT be executed in a context that DOES NOT permit sleeping.

Assumptions:

None

Side-Effects:

None

Blocking:

Yes when configured to operate in synchronous mode.

Reentrant:

No

Thread-safe:

Yes

Parameters

in	*instanceHandle*	Instance handle.
in	*pRsaEncryptCb*	Pointer to callback function to be invoked when the operation is complete. If this is set to a NULL value the function will operate synchronously.
in	*pCallbackTag*	Opaque User Data for this specific call. Will be returned unchanged in the callback.
in	*pEncryptOpData*	Structure containing all the data needed to perform the RSA encryption operation. The client code allocates the memory for this structure. This component takes ownership of the memory until it is returned in the callback.
out	*pOutputData*	Pointer to structure into which the result of the RSA encryption primitive is written. The client MUST allocate this memory. The data pointed to is an integer in big-endian order. The value will be between 0 and the modulus n - 1. On invocation the callback function will contain this parameter in the pOut parameter.

Return values

CPA_STATUS_SUCCESS	Function executed successfully.
CPA_STATUS_FAIL	Function failed.
CPA_STATUS_RETRY	Resubmit the request.
CPA_STATUS_INVALID_PARAM	Invalid parameter passed in.
CPA_STATUS_RESOURCE	Error related to system resources.
CPA_STATUS_RESTARTING	API implementation is restarting. Resubmit the request.
CPA_STATUS_UNSUPPORTED	Function is not supported.

Precondition

The component has been initialized via cpaCyStartInstance function.

Postcondition

None

Note

When pRsaEncryptCb is non-NULL an asynchronous callback of type is generated in response to this function call. Any errors generated during processing are reported as part of the callback status code. For optimal performance, data pointers SHOULD be 8-byte aligned.

See also

CpaCyGenFlatBufCbFunc CpaCyRsaEncryptOpData cpaCyRsaGenKey() cpaCyRsaDecrypt()

4.10.4.3 cpaCyRsaDecrypt()

```
CpaStatus cpaCyRsaDecrypt (
            const CpaInstanceHandle instanceHandle,
            const CpaCyGenFlatBufCbFunc pRsaDecryptCb,
            void * pCallbackTag,
            const CpaCyRsaDecryptOpData * pDecryptOpData,
            CpaFlatBuffer * pOutputData )
```

Perform the RSA decrypt (or sign) primitive operation on the input data.

Description:

This function will perform an RSA decryption primitive operation on the input data using the specified RSA private key. As the RSA decryption primitive and signing primitive operations are mathematically identical this function may also be used to perform an RSA signing primitive operation.

Context:

When called as an asynchronous function it cannot sleep. It can be executed in a context that does not permit sleeping. When called as a synchronous function it may sleep. It MUST NOT be executed in a context that DOES NOT permit sleeping.

Assumptions:

None

Side-Effects:

None

Blocking:

Yes when configured to operate in synchronous mode.

Reentrant:

No

Thread-safe:

Yes

Parameters

`in`	*instanceHandle*	Instance handle.
`in`	*pRsaDecryptCb*	Pointer to callback function to be invoked when the operation is complete. If this is set to a NULL value the function will operate synchronously.
`in`	*pCallbackTag*	Opaque User Data for this specific call. Will be returned unchanged in the callback.
`in`	*pDecryptOpData*	Structure containing all the data needed to perform the RSA decrypt operation. The client code allocates the memory for this structure. This component takes ownership of the memory until it is returned in the callback.
`out`	*pOutputData*	Pointer to structure into which the result of the RSA decryption primitive is written. The client MUST allocate this memory. The data pointed to is an integer in big-endian order. The value will be between 0 and the modulus n - 1. On invocation the callback function will contain this parameter in the pOut parameter.

Return values

CPA_STATUS_SUCCESS	Function executed successfully.
CPA_STATUS_FAIL	Function failed.
CPA_STATUS_RETRY	Resubmit the request.
CPA_STATUS_INVALID_PARAM	Invalid parameter passed in.
CPA_STATUS_RESOURCE	Error related to system resources.
CPA_STATUS_RESTARTING	API implementation is restarting. Resubmit the request.
CPA_STATUS_UNSUPPORTED	Function is not supported.

Precondition

The component has been initialized via cpaCyStartInstance function.

Postcondition

None

Note

When pRsaDecryptCb is non-NULL an asynchronous callback is generated in response to this function call. Any errors generated during processing are reported as part of the callback status code. For optimal performance, data pointers SHOULD be 8-byte aligned.

See also

CpaCyRsaDecryptOpData, CpaCyGenFlatBufCbFunc, cpaCyRsaGenKey(), cpaCyRsaEncrypt()

4.10.4.4 cpaCyRsaQueryStats()

```
CpaStatus CPA_DEPRECATED cpaCyRsaQueryStats (
             const CpaInstanceHandle instanceHandle,
             struct _CpaCyRsaStats * pRsaStats )
```

Query statistics for a specific RSA instance.

Deprecated As of v1.3 of the Crypto API, this function has been deprecated, replaced by cpaCyRsaQueryStats64().

Description:

This function will query a specific instance for RSA statistics. The user MUST allocate the CpaCyRsaStats structure and pass the reference to that into this function call. This function will write the statistic results into the passed in CpaCyRsaStats structure.

Note: statistics returned by this function do not interrupt current data processing and as such can be slightly out of sync with operations that are in progress during the statistics retrieval process.

Context:

This is a synchronous function and it can sleep. It MUST NOT be executed in a context that DOES NOT permit sleeping.

Assumptions:

None

Side-Effects:

None

Blocking:

This function is synchronous and blocking.

Reentrant:

No

Thread-safe:

Yes

Parameters

in	*instanceHandle*	Instance handle.
out	*pRsaStats*	Pointer to memory into which the statistics will be written.

Return values

CPA_STATUS_SUCCESS	Function executed successfully.
CPA_STATUS_FAIL	Function failed.
CPA_STATUS_INVALID_PARAM	Invalid parameter passed in.
CPA_STATUS_RESOURCE	Error related to system resources.
CPA_STATUS_RESTARTING	API implementation is restarting. Resubmit the request.
CPA_STATUS_UNSUPPORTED	Function is not supported.

Precondition

Component has been initialized.

Postcondition

None

Note

This function operates in a synchronous manner and no asynchronous callback will be generated.

See also

CpaCyRsaStats

4.10.4.5 cpaCyRsaQueryStats64()

```
CpaStatus cpaCyRsaQueryStats64 (
            const CpaInstanceHandle instanceHandle,
            CpaCyRsaStats64 * pRsaStats )
```

Query statistics (64-bit version) for a specific RSA instance.

Description:

This function will query a specific instance for RSA statistics. The user MUST allocate the CpaCyRsaStats64 structure and pass the reference to that into this function call. This function will write the statistic results into the passed in CpaCyRsaStats64 structure.

Note: statistics returned by this function do not interrupt current data processing and as such can be slightly out of sync with operations that are in progress during the statistics retrieval process.

Context:

This is a synchronous function and it can sleep. It MUST NOT be executed in a context that DOES NOT permit sleeping.

Assumptions:

None

Side-Effects:

None

Blocking:

This function is synchronous and blocking.

Reentrant:

No

Thread-safe:

Yes

Parameters

in	*instanceHandle*	Instance handle.
out	*pRsaStats*	Pointer to memory into which the statistics will be written.

Return values

CPA_STATUS_SUCCESS	Function executed successfully.
CPA_STATUS_FAIL	Function failed.
CPA_STATUS_INVALID_PARAM	Invalid parameter passed in.
CPA_STATUS_RESOURCE	Error related to system resources.
CPA_STATUS_RESTARTING	API implementation is restarting. Resubmit the request.
CPA_STATUS_UNSUPPORTED	Function is not supported.

Precondition

Component has been initialized.

Postcondition

None

Note

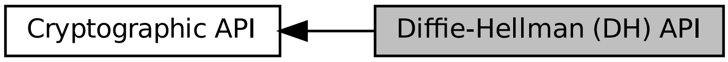
This function operates in a synchronous manner and no asynchronous callback will be generated.

See also

CpaCyRsaStats64

4.11 Diffie-Hellman (DH) API

Collaboration diagram for Diffie-Hellman (DH) API:



Data Structures

- struct _CpaCyDhPhase1KeyGenOpData
- struct _CpaCyDhPhase2SecretKeyGenOpData
- struct _CpaCyDhStats
- struct _CpaCyDhStats64

Typedefs

- typedef struct _CpaCyDhPhase1KeyGenOpData CpaCyDhPhase1KeyGenOpData
- typedef struct _CpaCyDhPhase2SecretKeyGenOpData CpaCyDhPhase2SecretKeyGenOpData
- typedef struct _CpaCyDhStats CPA_DEPRECATED
- typedef struct _CpaCyDhStats64 CpaCyDhStats64

Functions

- CpaStatus cpaCyDhKeyGenPhase1 (const CpaInstanceHandle instanceHandle, const CpaCyGenFlatBufCbFunc pDhPhase1Cb, void ∗pCallbackTag, const CpaCyDhPhase1KeyGenOpData ∗pPhase1KeyGenData, CpaFlatBuffer ∗pLocalOctetStringPV)
- CpaStatus cpaCyDhKeyGenPhase2Secret (const CpaInstanceHandle instanceHandle, const CpaCyGenFlatBufCbFunc pDhPhase2Cb, void ∗pCallbackTag, const CpaCyDhPhase2SecretKeyGenOpData ∗pPhase2SecretKeyGenData, CpaFlatBuffer ∗pOctetStringSecretKey)
- CpaStatus CPA_DEPRECATED cpaCyDhQueryStats (const CpaInstanceHandle instanceHandle, struct _CpaCyDhStats ∗pDhStats)
- CpaStatus cpaCyDhQueryStats64 (const CpaInstanceHandle instanceHandle, CpaCyDhStats64 ∗pDhStats)

4.11.1 Detailed Description

File: cpa_cy_dh.h

Description:

These functions specify the API for Public Key Encryption (Cryptography) operations for use with Diffie-Hellman algorithm.

Note

Large numbers are represented on the QuickAssist API as described in the Large Number API (Cryptographic Large Number API).

4.11.2 Typedef Documentation

4.11.2.1 CpaCyDhPhase1KeyGenOpData

```
typedef struct _CpaCyDhPhase1KeyGenOpData CpaCyDhPhase1KeyGenOpData
```

Diffie-Hellman Phase 1 Key Generation Data.

Description:

This structure lists the different items that are required in the cpaCyDhKeyGenPhase1 function. The client MUST allocate the memory for this structure. When the structure is passed into the function, ownership of the memory passes to the function. Ownership of the memory returns to the client when this structure is returned with the CpaCyDhPhase1KeyGenOpData structure.

Note

If the client modifies or frees the memory referenced in this structure after it has been submitted to the cpaCyDhKeyGenPhase1 function, and before it has been returned in the callback, undefined behavior will result. All values in this structure are required to be in Most Significant Byte first order, e.g. primeP.pData[0] = MSB.

4.11.2.2 CpaCyDhPhase2SecretKeyGenOpData

```
typedef struct _CpaCyDhPhase2SecretKeyGenOpData CpaCyDhPhase2SecretKeyGenOpData
```

Diffie-Hellman Phase 2 Secret Key Generation Data.

Description:

This structure lists the different items that required in the cpaCyDhKeyGenPhase2Secret function. The client MUST allocate the memory for this structure. When the structure is passed into the function, ownership of the memory passes to the function. Ownership of the memory returns to the client when this structure is returned with the callback.

Note

If the client modifies or frees the memory referenced in this structure after it has been submitted to the cpaCyDhKeyGenPhase2Secret function, and before it has been returned in the callback, undefined behavior will result. All values in this structure are required to be in Most Significant Byte first order, e.g. primeP.pData[0] = MSB.

4.11.2.3 CPA_DEPRECATED

```
typedef struct _CpaCyDhStats CPA_DEPRECATED
```

Diffie-Hellman Statistics.

Deprecated As of v1.3 of the Crypto API, this structure has been deprecated, replaced by CpaCyDhStats64.

Description:

This structure contains statistics on the Diffie-Hellman operations. Statistics are set to zero when the component is initialized, and are collected per instance.

4.11.2.4 CpaCyDhStats64

```
typedef struct _CpaCyDhStats64 CpaCyDhStats64
```

Diffie-Hellman Statistics (64-bit version).

Description:

This structure contains the 64-bit version of the statistics on the Diffie-Hellman operations. Statistics are set to zero when the component is initialized, and are collected per instance.

4.11.3 Function Documentation

4.11.3.1 cpaCyDhKeyGenPhase1()

```
CpaStatus cpaCyDhKeyGenPhase1 (
            const CpaInstanceHandle instanceHandle,
            const CpaCyGenFlatBufCbFunc pDhPhase1Cb,
            void * pCallbackTag,
            const CpaCyDhPhase1KeyGenOpData * pPhase1KeyGenData,
            CpaFlatBuffer * pLocalOctetStringPV )
```

Function to implement Diffie-Hellman phase 1 operations.

Description:

This function may be used to implement the Diffie-Hellman phase 1 operations as defined in the PKCS #3 standard. It may be used to generate the the (local) octet string public value (PV) key. The prime number sizes specified in RFC 2409, 4306, and part of RFC 3526 are supported (bit size 6144 from RFC 3536 is not supported).

Context:

When called as an asynchronous function it cannot sleep. It can be executed in a context that does not permit sleeping. When called as a synchronous function it may sleep. It MUST NOT be executed in a context that DOES NOT permit sleeping.

Assumptions:

None

Side-Effects:

None

Blocking:

Yes when configured to operate in synchronous mode.

Reentrant:

No

Thread-safe:

Yes

Parameters

in	*instanceHandle*	Instance handle.
in	*pDhPhase1Cb*	Pointer to a callback function to be invoked when the operation is complete. If the pointer is set to a NULL value the function will operate synchronously.
in	*pCallbackTag*	Opaque User Data for this specific call. Will be returned unchanged in the callback
in	*pPhase1KeyGenData*	Structure containing all the data needed to perform the DH Phase 1 key generation operation. The client code allocates the memory for this structure. This component takes ownership of the memory until it is returned in the callback.
out	*pLocalOctetStringPV*	Pointer to memory allocated by the client into which the (local) octet string Public Value (PV) will be written. This value needs to be sent to the remote entity with which Diffie-Hellman is negotiating. The size of this buffer in bytes (as represented by the dataLenInBytes field) MUST be at least big enough to store the public value, which may have a bit length up to that of pPrimeP. On invocation the callback function will contain this parameter in the pOut parameter.

Return values

CPA_STATUS_SUCCESS	Function executed successfully.
CPA_STATUS_FAIL	Function failed.
CPA_STATUS_RETRY	Resubmit the request.
CPA_STATUS_INVALID_PARAM	Invalid parameter passed in.
CPA_STATUS_RESOURCE	Error related to system resources.

Return values

CPA_STATUS_RESTARTING	API implementation is restarting. Resubmit the request.
CPA_STATUS_UNSUPPORTED	Function is not supported.

Precondition

The component has been initialized via cpaCyStartInstance function.

Postcondition

None

Note

When pDhPhase1Cb is non-NULL an asynchronous callback of type CpaCyGenFlatBufCbFunc is generated in response to this function call. Any errors generated during processing are reported in the structure returned in the callback.

See also

CpaCyGenFlatBufCbFunc, CpaCyDhPhase1KeyGenOpData

4.11.3.2 cpaCyDhKeyGenPhase2Secret()

```
CpaStatus cpaCyDhKeyGenPhase2Secret (
            const CpaInstanceHandle instanceHandle,
            const CpaCyGenFlatBufCbFunc pDhPhase2Cb,
            void * pCallbackTag,
            const CpaCyDhPhase2SecretKeyGenOpData * pPhase2SecretKeyGenData,
            CpaFlatBuffer * pOctetStringSecretKey )
```

Function to implement Diffie-Hellman phase 2 operations.

Description:

This function may be used to implement the Diffie-Hellman phase 2 operation as defined in the PKCS #3 standard. It may be used to generate the Diffie-Hellman shared secret key.

Context:

When called as an asynchronous function it cannot sleep. It can be executed in a context that does not permit sleeping. When called as a synchronous function it may sleep. It MUST NOT be executed in a context that DOES NOT permit sleeping.

Assumptions:

None

Side-Effects:

None

Blocking:

Yes when configured to operate in synchronous mode.

Reentrant:

No

Thread-safe:

Yes

Parameters

in	*instanceHandle*	Instance handle.
in	*pDhPhase2Cb*	Pointer to a callback function to be invoked when the operation is complete. If the pointer is set to a NULL value the function will operate synchronously.
in	*pCallbackTag*	Opaque User Data for this specific call. Will be returned unchanged in the callback.
in	*pPhase2SecretKeyGenData*	Structure containing all the data needed to perform the DH Phase 2 secret key generation operation. The client code allocates the memory for this structure. This component takes ownership of the memory until it is returned in the callback.
out	*pOctetStringSecretKey*	Pointer to memory allocated by the client into which the octet string secret key will be written. The size of this buffer in bytes (as represented by the dataLenInBytes field) MUST be at least big enough to store the public value, which may have a bit length up to that of pPrimeP. On invocation the callback function will contain this parameter in the pOut parameter.

Return values

CPA_STATUS_SUCCESS	Function executed successfully.
CPA_STATUS_FAIL	Function failed.
CPA_STATUS_RETRY	Resubmit the request.
CPA_STATUS_INVALID_PARAM	Invalid parameter passed in.
CPA_STATUS_RESOURCE	Error related to system resources.
CPA_STATUS_RESTARTING	API implementation is restarting. Resubmit the request.
CPA_STATUS_UNSUPPORTED	Function is not supported.

Precondition

The component has been initialized via cpaCyStartInstance function.

Postcondition

None

Note

When pDhPhase2Cb is non-NULL an asynchronous callback of type CpaCyGenFlatBufCbFunc is generated in response to this function call. Any errors generated during processing are reported in the structure returned in the callback.

See also

CpaCyGenFlatBufCbFunc, CpaCyDhPhase2SecretKeyGenOpData

4.11.3.3 cpaCyDhQueryStats()

```
CpaStatus CPA_DEPRECATED cpaCyDhQueryStats (
            const CpaInstanceHandle instanceHandle,
            struct _CpaCyDhStats * pDhStats )
```

Query statistics for Diffie-Hellman operations

Deprecated As of v1.3 of the Crypto API, this function has been deprecated, replaced by cpaCyDhQueryStats64().

Description:

This function will query a specific Instance handle for Diffie- Hellman statistics. The user MUST allocate the CpaCyDhStats structure and pass the reference to that structure into this function call. This function writes the statistic results into the passed in CpaCyDhStats structure.

Note: statistics returned by this function do not interrupt current data processing and as such can be slightly out of sync with operations that are in progress during the statistics retrieval process.

Context:

This is a synchronous function and it can sleep. It MUST NOT be executed in a context that DOES NOT permit sleeping.

Assumptions:

None

Side-Effects:

None

Reentrant:

No

Thread-safe:

Yes

Parameters

in	*instanceHandle*	Instance handle.
out	*pDhStats*	Pointer to memory into which the statistics will be written.

Return values

CPA_STATUS_SUCCESS	Function executed successfully.
CPA_STATUS_FAIL	Function failed.
CPA_STATUS_INVALID_PARAM	Invalid parameter passed in.
CPA_STATUS_RESOURCE	Error related to system resources.
CPA_STATUS_RESTARTING	API implementation is restarting. Resubmit the request.
CPA_STATUS_UNSUPPORTED	Function is not supported.

Precondition

Component has been initialized.

Postcondition

None

Note

This function operates in a synchronous manner and no asynchronous callback will be generated.

See also

CpaCyDhStats

4.11.3.4 cpaCyDhQueryStats64()

```
CpaStatus cpaCyDhQueryStats64 (
            const CpaInstanceHandle instanceHandle,
            CpaCyDhStats64 * pDhStats )
```

Query statistics (64-bit version) for Diffie-Hellman operations

Description:

This function will query a specific Instance handle for the 64-bit version of the Diffie-Hellman statistics. The user MUST allocate the CpaCyDhStats64 structure and pass the reference to that structure into this function call. This function writes the statistic results into the passed in CpaCyDhStats64 structure.

Note: statistics returned by this function do not interrupt current data processing and as such can be slightly out of sync with operations that are in progress during the statistics retrieval process.

Context:

This is a synchronous function and it can sleep. It MUST NOT be executed in a context that DOES NOT permit sleeping.

Assumptions:

None

Side-Effects:

None

Reentrant:

No

Thread-safe:

Yes

Parameters

in	*instanceHandle*	Instance handle.
out	*pDhStats*	Pointer to memory into which the statistics will be written.

Return values

CPA_STATUS_SUCCESS	Function executed successfully.
CPA_STATUS_FAIL	Function failed.
CPA_STATUS_INVALID_PARAM	Invalid parameter passed in.
CPA_STATUS_RESOURCE	Error related to system resources.
CPA_STATUS_RESTARTING	API implementation is restarting. Resubmit the request.
CPA_STATUS_UNSUPPORTED	Function is not supported.

Precondition

Component has been initialized.

Postcondition

None

Note

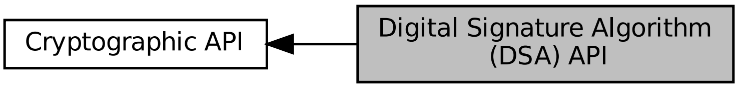
This function operates in a synchronous manner and no asynchronous callback will be generated.

See also

CpaCyDhStats64

4.12 Digital Signature Algorithm (DSA) API

Collaboration diagram for Digital Signature Algorithm (DSA) API:



Data Structures

- struct _CpaCyDsaPParamGenOpData
- struct _CpaCyDsaGParamGenOpData
- struct _CpaCyDsaYParamGenOpData
- struct _CpaCyDsaRSignOpData
- struct _CpaCyDsaSSignOpData
- struct _CpaCyDsaRSSignOpData
- struct _CpaCyDsaVerifyOpData
- struct _CpaCyDsaStats
- struct _CpaCyDsaStats64

Typedefs

- typedef struct _CpaCyDsaPParamGenOpData CpaCyDsaPParamGenOpData
- typedef struct _CpaCyDsaGParamGenOpData CpaCyDsaGParamGenOpData
- typedef struct _CpaCyDsaYParamGenOpData CpaCyDsaYParamGenOpData
- typedef struct _CpaCyDsaRSignOpData CpaCyDsaRSignOpData
- typedef struct _CpaCyDsaSSignOpData CpaCyDsaSSignOpData
- typedef struct _CpaCyDsaRSSignOpData CpaCyDsaRSSignOpData
- typedef struct _CpaCyDsaVerifyOpData CpaCyDsaVerifyOpData
- typedef struct _CpaCyDsaStats CPA_DEPRECATED
- typedef struct _CpaCyDsaStats64 CpaCyDsaStats64
- typedef void(∗ CpaCyDsaGenCbFunc) (void ∗pCallbackTag, CpaStatus status, void ∗pOpData, CpaBoolean protocolStatus, CpaFlatBuffer ∗pOut)
- typedef void(∗ CpaCyDsaRSSignCbFunc) (void ∗pCallbackTag, CpaStatus status, void ∗pOpData, CpaBoolean protocolStatus, CpaFlatBuffer ∗pR, CpaFlatBuffer ∗pS)
- typedef void(∗ CpaCyDsaVerifyCbFunc) (void ∗pCallbackTag, CpaStatus status, void ∗pOpData, CpaBoolean verifyStatus)

Functions

- CpaStatus cpaCyDsaGenPParam (const CpaInstanceHandle instanceHandle, const CpaCyDsaGenCbFunc pCb, void ∗pCallbackTag, const CpaCyDsaPParamGenOpData ∗pOpData, CpaBoolean ∗pProtocolStatus, CpaFlatBuffer ∗pP)
- CpaStatus cpaCyDsaGenGParam (const CpaInstanceHandle instanceHandle, const CpaCyDsaGenCbFunc pCb, void ∗pCallbackTag, const CpaCyDsaGParamGenOpData ∗pOpData, CpaBoolean ∗pProtocolStatus, CpaFlatBuffer ∗pG)
- CpaStatus cpaCyDsaGenYParam (const CpaInstanceHandle instanceHandle, const CpaCyDsaGenCbFunc pCb, void ∗pCallbackTag, const CpaCyDsaYParamGenOpData ∗pOpData, CpaBoolean ∗pProtocolStatus, CpaFlatBuffer ∗pY)
- CpaStatus cpaCyDsaSignR (const CpaInstanceHandle instanceHandle, const CpaCyDsaGenCbFunc pCb, void ∗pCallbackTag, const CpaCyDsaRSignOpData ∗pOpData, CpaBoolean ∗pProtocolStatus, CpaFlatBuffer ∗pR)
- CpaStatus cpaCyDsaSignS (const CpaInstanceHandle instanceHandle, const CpaCyDsaGenCbFunc pCb, void ∗pCallbackTag, const CpaCyDsaSSignOpData ∗pOpData, CpaBoolean ∗pProtocolStatus, CpaFlatBuffer ∗pS)
- CpaStatus cpaCyDsaSignRS (const CpaInstanceHandle instanceHandle, const CpaCyDsaRSSignCbFunc pCb, void ∗pCallbackTag, const CpaCyDsaRSSignOpData ∗pOpData, CpaBoolean ∗pProtocolStatus, CpaFlatBuffer ∗pR, CpaFlatBuffer ∗pS)
- CpaStatus cpaCyDsaVerify (const CpaInstanceHandle instanceHandle, const CpaCyDsaVerifyCbFunc pCb, void ∗pCallbackTag, const CpaCyDsaVerifyOpData ∗pOpData, CpaBoolean ∗pVerifyStatus)
- CpaStatus CPA_DEPRECATED cpaCyDsaQueryStats (const CpaInstanceHandle instanceHandle, struct _CpaCyDsaStats ∗pDsaStats)
- CpaStatus cpaCyDsaQueryStats64 (const CpaInstanceHandle instanceHandle, CpaCyDsaStats64 ∗pDsaStats)

4.12.1 Detailed Description

File: cpa_cy_dsa.h

Description:

These functions specify the API for Public Key Encryption (Cryptography) Digital Signature Algorithm (DSA) operations.

Support is provided for FIPS PUB 186-2 with Change Notice 1 specification, and optionally for FIPS PUB 186-3. If an implementation does not support FIPS PUB 186-3, then the corresponding functions may return a status of CPA_STATUS_FAIL.

Support for FIPS PUB 186-2 with Change Notice 1 implies supporting the following choice for the pair L and N:

- L = 1024, N = 160

Support for FIPS PUB 186-3 implies supporting the following choices for the pair L and N:

- L = 1024, N = 160
- L = 2048, N = 224
- L = 2048, N = 256
- L = 3072, N = 256

Only the modular math aspects of DSA parameter generation and message signature generation and verification are implemented here. For full DSA support, this DSA API SHOULD be used in conjunction with other parts of this overall Cryptographic API. In particular the Symmetric functions (for hashing), the Random Number Generation functions, and the Prime Number Test functions will be required.

Note

Large numbers are represented on the QuickAssist API as described in the Large Number API (Cryptographic Large Number API).

4.12.2 Typedef Documentation

4.12.2.1 CpaCyDsaPParamGenOpData

`typedef struct _CpaCyDsaPParamGenOpData CpaCyDsaPParamGenOpData`

DSA P Parameter Generation Operation Data.

Description:

This structure contains the operation data for the cpaCyDsaGenPParam function. The client MUST allocate the memory for this structure and the items pointed to by this structure. When the structure is passed into the function, ownership of the memory passes to the function. Ownership of the memory returns to the client when this structure is returned in the callback function.

For optimal performance all data buffers SHOULD be 8-byte aligned.

All values in this structure are required to be in Most Significant Byte first order, e.g. X.pData[0] = MSB.

Note

If the client modifies or frees the memory referenced in this structure after it has been submitted to the cpaCyDsaGenPParam function, and before it has been returned in the callback, undefined behavior will result.

See also

cpaCyDsaGenPParam()

4.12.2.2 CpaCyDsaGParamGenOpData

`typedef struct _CpaCyDsaGParamGenOpData CpaCyDsaGParamGenOpData`

DSA G Parameter Generation Operation Data.

Description:

This structure contains the operation data for the cpaCyDsaGenGParam function. The client MUST allocate the memory for this structure and the items pointed to by this structure. When the structure is passed into the function, ownership of the memory passes to the function. Ownership of the memory returns to the client when this structure is returned in the callback function.

All values in this structure are required to be in Most Significant Byte first order, e.g. P.pData[0] = MSB.

All numbers MUST be stored in big-endian order.

Note

If the client modifies or frees the memory referenced in this structure after it has been submitted to the cpaCyDsaGenGParam function, and before it has been returned in the callback, undefined behavior will result.

See also

cpaCyDsaGenGParam()

4.12.2.3 CpaCyDsaYParamGenOpData

```
typedef struct _CpaCyDsaYParamGenOpData CpaCyDsaYParamGenOpData
```

DSA Y Parameter Generation Operation Data.

Description:

This structure contains the operation data for the cpaCyDsaGenYParam function. The client MUST allocate the memory for this structure and the items pointed to by this structure. When the structure is passed into the function, ownership of the memory passes to the function. Ownership of the memory returns to the client when this structure is returned in the callback function.

For optimal performance all data SHOULD be 8-byte aligned.

All values in this structure are required to be in Most Significant Byte first order, e.g. P.pData[0] = MSB.

Note

If the client modifies or frees the memory referenced in this structure after it has been submitted to the cpaCyDsaGenYParam function, and before it has been returned in the callback, undefined behavior will result.

See also

cpaCyDsaGenYParam()

4.12.2.4 CpaCyDsaRSignOpData

```
typedef struct _CpaCyDsaRSignOpData CpaCyDsaRSignOpData
```

DSA R Sign Operation Data.

Description:

This structure contains the operation data for the cpaCyDsaSignR function. The client MUST allocate the memory for this structure and the items pointed to by this structure. When the structure is passed into the function, ownership of the memory passes to the function. Ownership of the memory returns to the client when this structure is returned in the callback function.

For optimal performance all data SHOULD be 8-byte aligned.

All values in this structure are required to be in Most Significant Byte first order, e.g. P.pData[0] = MSB.

Note

If the client modifies or frees the memory referenced in this structure after it has been submitted to the cpaCyDsaSignR function, and before it has been returned in the callback, undefined behavior will result.

See also

cpaCyDsaSignR()

4.12.2.5 CpaCyDsaSSignOpData

```
typedef struct _CpaCyDsaSSignOpData CpaCyDsaSSignOpData
```

DSA S Sign Operation Data.

Description:

This structure contains the operation data for the cpaCyDsaSignS function. The client MUST allocate the memory for this structure and the items pointed to by this structure. When the structure is passed into the function, ownership of the memory passes to the function. Ownership of the memory returns to the client when this structure is returned in the callback function.

For optimal performance all data SHOULD be 8-byte aligned.

All values in this structure are required to be in Most Significant Byte first order, e.g. Q.pData[0] = MSB.

Note

If the client modifies or frees the memory referenced in this structure after it has been submitted to the cpaCyDsaSignS function, and before it has been returned in the callback, undefined behavior will result.

See also

cpaCyDsaSignS()

4.12.2.6 CpaCyDsaRSSignOpData

```
typedef struct _CpaCyDsaRSSignOpData CpaCyDsaRSSignOpData
```

DSA R & S Sign Operation Data.

Description:

This structure contains the operation data for the cpaCyDsaSignRS function. The client MUST allocate the memory for this structure and the items pointed to by this structure. When the structure is passed into the function, ownership of the memory passes to the function. Ownership of the memory returns to the client when this structure is returned in the callback function.

For optimal performance all data SHOULD be 8-byte aligned.

All values in this structure are required to be in Most Significant Byte first order, e.g. P.pData[0] = MSB.

Note

If the client modifies or frees the memory referenced in this structure after it has been submitted to the cpaCyDsaSignRS function, and before it has been returned in the callback, undefined behavior will result.

See also

cpaCyDsaSignRS()

4.12.2.7 CpaCyDsaVerifyOpData

```
typedef struct _CpaCyDsaVerifyOpData CpaCyDsaVerifyOpData
```

DSA Verify Operation Data.

Description:

This structure contains the operation data for the cpaCyDsaVerify function. The client MUST allocate the memory for this structure and the items pointed to by this structure. When the structure is passed into the function, ownership of the memory passes to the function. Ownership of the memory returns to the client when this structure is returned in the callback function.

For optimal performance all data SHOULD be 8-byte aligned.

All values in this structure are required to be in Most Significant Byte first order, e.g. P.pData[0] = MSB.

Note

If the client modifies or frees the memory referenced in this structure after it has been submitted to the cpaCyDsaVerify function, and before it has been returned in the callback, undefined behavior will result.

See also

cpaCyDsaVerify()

4.12.2.8 CPA_DEPRECATED

```
typedef struct _CpaCyDsaStats CPA_DEPRECATED
```

Cryptographic DSA Statistics.

Deprecated As of v1.3 of the Crypto API, this structure has been deprecated, replaced by CpaCyDsaStats64.

Description:

This structure contains statistics on the Cryptographic DSA operations. Statistics are set to zero when the component is initialized, and are collected per instance.

4.12.2.9 CpaCyDsaStats64

```
typedef struct _CpaCyDsaStats64 CpaCyDsaStats64
```

Cryptographic DSA Statistics (64-bit version).

Description:

This structure contains 64-bit version of the statistics on the Cryptographic DSA operations. Statistics are set to zero when the component is initialized, and are collected per instance.

4.12.2.10 CpaCyDsaGenCbFunc

```
typedef void(* CpaCyDsaGenCbFunc) (void *pCallbackTag, CpaStatus status, void *pOpData,
CpaBoolean protocolStatus, CpaFlatBuffer *pOut)
```

Definition of a generic callback function invoked for a number of the DSA API functions..

Description:

This is the prototype for the cpaCyDsaGenCbFunc callback function.

Context:

This callback function can be executed in a context that DOES NOT permit sleeping to occur.

Assumptions:

None

Side-Effects:

None

Reentrant:

No

Thread-safe:

Yes

Parameters

in	*pCallbackTag*	User-supplied value to help identify request.
in	*status*	Status of the operation. Valid values are CPA_STATUS_SUCCESS, CPA_STATUS_FAIL and CPA_STATUS_UNSUPPORTED.
in	*pOpData*	Opaque pointer to Operation data supplied in request.
in	*protocolStatus*	The result passes/fails the DSA protocol related checks.
in	*pOut*	Output data from the request.

Return values

None	

Precondition

Component has been initialized.

Postcondition

None

Note

None

See also

cpaCyDsaGenPParam() cpaCyDsaGenGParam() cpaCyDsaSignR() cpaCyDsaSignS()

Definition at line 581 of file cpa_cy_dsa.h.

4.12.2.11 CpaCyDsaRSSignCbFunc

```
typedef void(* CpaCyDsaRSSignCbFunc) (void *pCallbackTag, CpaStatus status, void *pOpData,
CpaBoolean protocolStatus, CpaFlatBuffer *pR, CpaFlatBuffer *pS)
```

Definition of callback function invoked for cpaCyDsaSignRS requests.

Description:

This is the prototype for the cpaCyDsaSignRS callback function, which will provide the DSA message signature r and s parameters.

Context:

This callback function can be executed in a context that DOES NOT permit sleeping to occur.

Assumptions:

None

Side-Effects:

None

Reentrant:

No

Thread-safe:

Yes

Parameters

in	*pCallbackTag*	User-supplied value to help identify request.
in	*status*	Status of the operation. Valid values are CPA_STATUS_SUCCESS, CPA_STATUS_FAIL and CPA_STATUS_UNSUPPORTED.
in	*pOpData*	Operation data pointer supplied in request.
in	*protocolStatus*	The result passes/fails the DSA protocol related checks.
in	*pR*	DSA message signature r.
in	*pS*	DSA message signature s.

Return values

None	

Precondition

Component has been initialized.

Postcondition

None

Note

None

See also

cpaCyDsaSignRS()

Definition at line 632 of file cpa_cy_dsa.h.

4.12.2.12 CpaCyDsaVerifyCbFunc

```
typedef void(* CpaCyDsaVerifyCbFunc) (void *pCallbackTag, CpaStatus status, void *pOpData,
CpaBoolean verifyStatus)
```

Definition of callback function invoked for cpaCyDsaVerify requests.

Description:

This is the prototype for the cpaCyDsaVerify callback function.

Context:

This callback function can be executed in a context that DOES NOT permit sleeping to occur.

Assumptions:

None

Side-Effects:

None

Reentrant:

No

Thread-safe:

Yes

Parameters

in	*pCallbackTag*	User-supplied value to help identify request.
in	*status*	Status of the operation. Valid values are CPA_STATUS_SUCCESS, CPA_STATUS_FAIL and CPA_STATUS_UNSUPPORTED.
in	*pOpData*	Operation data pointer supplied in request.
in	*verifyStatus*	The verification passed or failed.

Return values

None	

Precondition

Component has been initialized.

Postcondition

None

Note

None

See also

cpaCyDsaVerify()

Definition at line 679 of file cpa_cy_dsa.h.

4.12.3 Function Documentation

4.12.3.1 cpaCyDsaGenPParam()

```
CpaStatus cpaCyDsaGenPParam (
            const CpaInstanceHandle instanceHandle,
            const CpaCyDsaGenCbFunc pCb,
            void * pCallbackTag,
            const CpaCyDsaPParamGenOpData * pOpData,
            CpaBoolean * pProtocolStatus,
            CpaFlatBuffer * pP )
```

Generate DSA P Parameter.

Description:

```
This function performs FIPS 186-3 Appendix A.1.1.2 steps 11.4 and 11.5,
and part of step 11.7:

    11.4. c = X mod 2q.
    11.5. p = X - (c - 1).
    11.7. Test whether or not p is prime as specified in Appendix C.3.
          [Note that a GCD test against ~1400 small primes is performed
          on p to eliminate ~94% of composites - this is NOT a "robust"
          primality test, as specified in Appendix C.3.]

The protocol status, returned in the callback function as parameter
protocolStatus (or, in the case of synchronous invocation, in the
parameter *pProtocolStatus) is used to indicate whether the value p is
in the right range and has passed the limited primality test.

Specifically, (protocolStatus == CPA_TRUE) means p is in the right range
and  SHOULD be subjected to a robust primality test as specified in
FIPS 186-3 Appendix C.3 (for example, 40 rounds of Miller-Rabin).
Meanwhile, (protocolStatus == CPA_FALSE) means p is either composite,
or p < 2^(L-1), in which case the value of p gets set to zero.
```

Context:

When called as an asynchronous function it cannot sleep. It can be executed in a context that does not permit sleeping. When called as a synchronous function it may sleep. It MUST NOT be executed in a context that DOES NOT permit sleeping.

Assumptions:

None

Side-Effects:

None

Blocking:

Yes when configured to operate in synchronous mode.

Reentrant:

No

Thread-safe:

Yes

Parameters

in	*instanceHandle*	Instance handle.
in	*pCb*	Callback function pointer. If this is set to a NULL value the function will operate synchronously.
in	*pCallbackTag*	User-supplied value to help identify request.
in	*pOpData*	Structure containing all the data needed to perform the operation. The client code allocates the memory for this structure. This component takes ownership of the memory until it is returned in the callback.
out	*pProtocolStatus*	The result passes/fails the DSA protocol related checks.
out	*pP*	Candidate for DSA parameter p, p odd and 2^(L-1) < p < X On invocation the callback function will contain this parameter in the pOut parameter.

Return values

CPA_STATUS_SUCCESS	Function executed successfully.
CPA_STATUS_FAIL	Function failed.
CPA_STATUS_RETRY	Resubmit the request.
CPA_STATUS_INVALID_PARAM	Invalid parameter passed in.
CPA_STATUS_RESOURCE	Error related to system resources.
CPA_STATUS_RESTARTING	API implementation is restarting. Resubmit the request.
CPA_STATUS_UNSUPPORTED	Function is not supported.

Precondition

The component has been initialized.

Postcondition

None

Note

When pCb is non-NULL an asynchronous callback of type CpaCyDsaPParamGenCbFunc is generated in response to this function call. For optimal performance, data pointers SHOULD be 8-byte aligned.

See also

CpaCyDsaPParamGenOpData, CpaCyDsaGenCbFunc

4.12.3.2 cpaCyDsaGenGParam()

```
CpaStatus cpaCyDsaGenGParam (
            const CpaInstanceHandle instanceHandle,
            const CpaCyDsaGenCbFunc pCb,
            void * pCallbackTag,
            const CpaCyDsaGParamGenOpData * pOpData,
            CpaBoolean * pProtocolStatus,
            CpaFlatBuffer * pG )
```

Generate DSA G Parameter.

Description:

This function performs FIPS 186-3 Appendix A.2.1, steps 1 and 3, and part of step 4:

```
1. e = (p - 1)/q.
3. Set g = h^e mod p.
4. If (g = 1), then go to step 2.
   Here, the implementation will check for g == 1, and return
   status accordingly.
```

The protocol status, returned in the callback function as parameter protocolStatus (or, in the case of synchronous invocation, in the parameter ∗pProtocolStatus) is used to indicate whether the value g is acceptable.

Specifically, (protocolStatus == CPA_TRUE) means g is acceptable. Meanwhile, (protocolStatus == CPA_FALSE) means g == 1, so a different value of h SHOULD be used to generate another value of g.

Context:

When called as an asynchronous function it cannot sleep. It can be executed in a context that does not permit sleeping. When called as a synchronous function it may sleep. It MUST NOT be executed in a context that DOES NOT permit sleeping.

Assumptions:

None

Side-Effects:

None

Blocking:

Yes when configured to operate in synchronous mode.

Reentrant:

No

Thread-safe:

Yes

Parameters

in	*instanceHandle*	Instance handle.
in	*pCb*	Callback function pointer. If this is set to a NULL value the function will operate synchronously.
in	*pCallbackTag*	User-supplied value to help identify request.
in	*pOpData*	Structure containing all the data needed to perform the operation. The client code allocates the memory for this structure. This component takes ownership of the memory until it is returned in the callback.
out	*pProtocolStatus*	The result passes/fails the DSA protocol related checks.
out	*pG*	g = h^((p-1)/q) mod p. On invocation the callback function will contain this parameter in the pOut parameter.

Return values

CPA_STATUS_SUCCESS	Function executed successfully.
CPA_STATUS_FAIL	Function failed.
CPA_STATUS_RETRY	Resubmit the request.
CPA_STATUS_INVALID_PARAM	Invalid parameter passed in.
CPA_STATUS_RESOURCE	Error related to system resources.
CPA_STATUS_RESTARTING	API implementation is restarting. Resubmit the request.
CPA_STATUS_UNSUPPORTED	Function is not supported.

Precondition

The component has been initialized via cpaCyStartInstance function.

Postcondition

None

Note

When pCb is non-NULL an asynchronous callback of type CpaCyDsaGParamGenCbFunc is generated in response to this function call. For optimal performance, data pointers SHOULD be 8-byte aligned.

See also

CpaCyDsaGParamGenOpData, CpaCyDsaGenCbFunc

4.12.3.3 cpaCyDsaGenYParam()

```
CpaStatus cpaCyDsaGenYParam (
            const CpaInstanceHandle instanceHandle,
            const CpaCyDsaGenCbFunc pCb,
            void * pCallbackTag,
            const CpaCyDsaYParamGenOpData * pOpData,
            CpaBoolean * pProtocolStatus,
            CpaFlatBuffer * pY )
```

Generate DSA Y Parameter.

Description:

```
This function performs modular exponentiation to generate y as
described in FIPS 186-3 section 4.1:
    y = g^x mod p
```

Context:

When called as an asynchronous function it cannot sleep. It can be executed in a context that does not permit sleeping. When called as a synchronous function it may sleep. It MUST NOT be executed in a context that DOES NOT permit sleeping.

Assumptions:

None

Side-Effects:

None

Blocking:

Yes when configured to operate in synchronous mode.

Reentrant:

No

Thread-safe:

Yes

Parameters

in	*instanceHandle*	Instance handle.
in	*pCb*	Callback function pointer. If this is set to a NULL value the function will operate synchronously.
in	*pCallbackTag*	User-supplied value to help identify request.
in	*pOpData*	Structure containing all the data needed to perform the operation. The client code allocates the memory for this structure. This component takes ownership of the memory until it is returned in the callback.
out	*pProtocolStatus*	The result passes/fails the DSA protocol related checks.
out	*pY*	y = g^x mod p∗ On invocation the callback function will contain this parameter in the pOut parameter.

Return values

CPA_STATUS_SUCCESS	Function executed successfully.
CPA_STATUS_FAIL	Function failed.
CPA_STATUS_RETRY	Resubmit the request.
CPA_STATUS_INVALID_PARAM	Invalid parameter passed in.
CPA_STATUS_RESOURCE	Error related to system resources.
CPA_STATUS_RESTARTING	API implementation is restarting. Resubmit the request.
CPA_STATUS_UNSUPPORTED	Function is not supported.

Precondition

The component has been initialized via cpaCyStartInstance function.

Postcondition

None

Note

When pCb is non-NULL an asynchronous callback of type CpaCyDsaYParamGenCbFunc is generated in response to this function call. For optimal performance, data pointers SHOULD be 8-byte aligned.

See also

CpaCyDsaYParamGenOpData, CpaCyDsaGenCbFunc

4.12.3.4 cpaCyDsaSignR()

```
CpaStatus cpaCyDsaSignR (
            const CpaInstanceHandle instanceHandle,
            const CpaCyDsaGenCbFunc pCb,
            void * pCallbackTag,
            const CpaCyDsaRSignOpData * pOpData,
            CpaBoolean * pProtocolStatus,
            CpaFlatBuffer * pR )
```

Generate DSA R Signature.

Description:

This function generates the DSA R signature as described in FIPS 186-3 Section 4.6: r = (g^k mod p) mod q

The protocol status, returned in the callback function as parameter protocolStatus (or, in the case of synchronous invocation, in the parameter *pProtocolStatus) is used to indicate whether the value r == 0.

Specifically, (protocolStatus == CPA_TRUE) means r != 0, while (protocolStatus == CPA_FALSE) means r == 0.

Generation of signature r does not depend on the content of the message being signed, so this operation can be done in advance for different values of k. Then once each message becomes available only the signature s needs to be generated.

Context:

When called as an asynchronous function it cannot sleep. It can be executed in a context that does not permit sleeping. When called as a synchronous function it may sleep. It MUST NOT be executed in a context that DOES NOT permit sleeping.

Assumptions:

None

Side-Effects:

None

Blocking:

Yes when configured to operate in synchronous mode.

Reentrant:

No

Thread-safe:

Yes

Parameters

in	*instanceHandle*	Instance handle.
in	*pCb*	Callback function pointer. If this is set to a NULL value the function will operate synchronously.
in	*pCallbackTag*	User-supplied value to help identify request.
in	*pOpData*	Structure containing all the data needed to perform the operation. The client code allocates the memory for this structure. This component takes ownership of the memory until it is returned in the callback.
out	*pProtocolStatus*	The result passes/fails the DSA protocol related checks.
out	*pR*	DSA message signature r. On invocation the callback function will contain this parameter in the pOut parameter.

Return values

CPA_STATUS_SUCCESS	Function executed successfully.
CPA_STATUS_FAIL	Function failed.
CPA_STATUS_RETRY	Resubmit the request.
CPA_STATUS_INVALID_PARAM	Invalid parameter passed in.
CPA_STATUS_RESOURCE	Error related to system resources.
CPA_STATUS_RESTARTING	API implementation is restarting. Resubmit the request.
CPA_STATUS_UNSUPPORTED	Function is not supported.

Precondition

The component has been initialized via cpaCyStartInstance function.

Postcondition

None

Note

When pCb is non-NULL an asynchronous callback of type CpaCyDsaRSignCbFunc is generated in response to this function call. For optimal performance, data pointers SHOULD be 8-byte aligned.

See also

CpaCyDsaRSignOpData, CpaCyDsaGenCbFunc, cpaCyDsaSignS(), cpaCyDsaSignRS()

4.12.3.5 cpaCyDsaSignS()

```
CpaStatus cpaCyDsaSignS (
            const CpaInstanceHandle instanceHandle,
            const CpaCyDsaGenCbFunc pCb,
            void * pCallbackTag,
            const CpaCyDsaSSignOpData * pOpData,
            CpaBoolean * pProtocolStatus,
            CpaFlatBuffer * pS )
```

Generate DSA S Signature.

Description:

This function generates the DSA S signature as described in FIPS 186-3 Section 4.6: s = (k^-1(z + xr)) mod q

Here, z = the leftmost min(N, outlen) bits of Hash(M). This function does not perform the SHA digest; z is computed by the caller and passed as a parameter in the pOpData field.

The protocol status, returned in the callback function as parameter protocolStatus (or, in the case of synchronous invocation, in the parameter *pProtocolStatus) is used to indicate whether the value s == 0.

Specifically, (protocolStatus == CPA_TRUE) means s != 0, while (protocolStatus == CPA_FALSE) means s == 0.

If signature r has been generated in advance, then this function can be used to generate the signature s once the message becomes available.

Context:

When called as an asynchronous function it cannot sleep. It can be executed in a context that does not permit sleeping. When called as a synchronous function it may sleep. It MUST NOT be executed in a context that DOES NOT permit sleeping.

Assumptions:

None

Side-Effects:

None

Blocking:

Yes when configured to operate in synchronous mode.

Reentrant:

No

Thread-safe:

Yes

Parameters

`in`	*instanceHandle*	Instance handle.
`in`	*pCb*	Callback function pointer. If this is set to a NULL value the function will operate synchronously.
`in`	*pCallbackTag*	User-supplied value to help identify request.
`in`	*pOpData*	Structure containing all the data needed to perform the operation. The client code allocates the memory for this structure. This component takes ownership of the memory until it is returned in the callback.
`out`	*pProtocolStatus*	The result passes/fails the DSA protocol related checks.
`out`	*pS*	DSA message signature s. On invocation the callback function will contain this parameter in the pOut parameter.

Return values

CPA_STATUS_SUCCESS	Function executed successfully.
CPA_STATUS_FAIL	Function failed.
CPA_STATUS_RETRY	Resubmit the request.
CPA_STATUS_INVALID_PARAM	Invalid parameter passed in.
CPA_STATUS_RESOURCE	Error related to system resources.
CPA_STATUS_RESTARTING	API implementation is restarting. Resubmit the request.
CPA_STATUS_UNSUPPORTED	Function is not supported.

Precondition

The component has been initialized via cpaCyStartInstance function.

Postcondition

None

Note

When pCb is non-NULL an asynchronous callback of type CpaCyDsaSSignCbFunc is generated in response to this function call. For optimal performance, data pointers SHOULD be 8-byte aligned.

See also

CpaCyDsaSSignOpData, CpaCyDsaGenCbFunc, cpaCyDsaSignR(), cpaCyDsaSignRS()

4.12.3.6 cpaCyDsaSignRS()

```
CpaStatus cpaCyDsaSignRS (
            const CpaInstanceHandle instanceHandle,
            const CpaCyDsaRSSignCbFunc pCb,
            void * pCallbackTag,
            const CpaCyDsaRSSignOpData * pOpData,
            CpaBoolean * pProtocolStatus,
            CpaFlatBuffer * pR,
            CpaFlatBuffer * pS )
```

Generate DSA R and S Signatures.

Description:

This function generates the DSA R and S signatures as described in FIPS 186-3 Section 4.6:

```
r = (g^k mod p) mod q
s = (k^-1(z + xr)) mod q
```

Here, z = the leftmost min(N, outlen) bits of Hash(M). This function does not perform the SHA digest; z is computed by the caller and passed as a parameter in the pOpData field.

The protocol status, returned in the callback function as parameter protocolStatus (or, in the case of synchronous invocation, in the parameter *pProtocolStatus) is used to indicate whether either of the values r or s are zero.

Specifically, (protocolStatus == CPA_TRUE) means neither is zero (i.e. (r != 0) && (s != 0)), while (protocolStatus == CPA_FALSE) means that at least one of r or s is zero (i.e. (r == 0) || (s == 0)).

Context:

When called as an asynchronous function it cannot sleep. It can be executed in a context that does not permit sleeping. When called as a synchronous function it may sleep. It MUST NOT be executed in a context that DOES NOT permit sleeping.

Assumptions:

None

Side-Effects:

None

Blocking:

Yes when configured to operate in synchronous mode.

Reentrant:

No

Thread-safe:

Yes

Parameters

in	*instanceHandle*	Instance handle.
in	*pCb*	Callback function pointer. If this is set to a NULL value the function will operate synchronously.
in	*pCallbackTag*	User-supplied value to help identify request.
in	*pOpData*	Structure containing all the data needed to perform the operation. The client code allocates the memory for this structure. This component takes ownership of the memory until it is returned in the callback.
out	*pProtocolStatus*	The result passes/fails the DSA protocol related checks.
out	*pR*	DSA message signature r.
out	*pS*	DSA message signature s.

Return values

CPA_STATUS_SUCCESS	Function executed successfully.
CPA_STATUS_FAIL	Function failed.
CPA_STATUS_RETRY	Resubmit the request.
CPA_STATUS_INVALID_PARAM	Invalid parameter passed in.
CPA_STATUS_RESOURCE	Error related to system resources.
CPA_STATUS_RESTARTING	API implementation is restarting. Resubmit the request.
CPA_STATUS_UNSUPPORTED	Function is not supported.

Precondition

The component has been initialized via cpaCyStartInstance function.

Postcondition

None

Note

When pCb is non-NULL an asynchronous callback of type CpaCyDsaRSSignCbFunc is generated in response to this function call. For optimal performance, data pointers SHOULD be 8-byte aligned.

See also

CpaCyDsaRSSignOpData, CpaCyDsaRSSignCbFunc, cpaCyDsaSignR(), cpaCyDsaSignS()

4.12.3.7 cpaCyDsaVerify()

```
CpaStatus cpaCyDsaVerify (
            const CpaInstanceHandle instanceHandle,
            const CpaCyDsaVerifyCbFunc pCb,
            void * pCallbackTag,
            const CpaCyDsaVerifyOpData * pOpData,
            CpaBoolean * pVerifyStatus )
```

Verify DSA R and S signatures.

Description:

This function performs FIPS 186-3 Section 4.7: w = (s')^-1 mod q u1 = (zw) mod q u2 = ((r')w) mod q v = (((g)^u1 (y)^u2) mod p) mod q

Here, z = the leftmost min(N, outlen) bits of Hash(M'). This function does not perform the SHA digest; z is computed by the caller and passed as a parameter in the pOpData field.

A response status of ok (verifyStatus == CPA_TRUE) means v = r'. A response status of not ok (verifyStatus == CPA_FALSE) means v != r'.

Context:

When called as an asynchronous function it cannot sleep. It can be executed in a context that does not permit sleeping. When called as a synchronous function it may sleep. It MUST NOT be executed in a context that DOES NOT permit sleeping.

Assumptions:

None

Side-Effects:

None

Blocking:

Yes when configured to operate in synchronous mode.

Reentrant:

No

Thread-safe:

Yes

Parameters

in	*instanceHandle*	Instance handle.
in	*pCb*	Callback function pointer. If this is set to a NULL value the function will operate synchronously.
in	*pCallbackTag*	User-supplied value to help identify request.
in	*pOpData*	Structure containing all the data needed to perform the operation. The client code allocates the memory for this structure. This component takes ownership of the memory until it is returned in the callback.
out	*pVerifyStatus*	The verification passed or failed.

Return values

CPA_STATUS_SUCCESS	Function executed successfully.
CPA_STATUS_FAIL	Function failed.
CPA_STATUS_RETRY	Resubmit the request.
CPA_STATUS_INVALID_PARAM	Invalid parameter passed in.
CPA_STATUS_RESOURCE	Error related to system resources.
CPA_STATUS_RESTARTING	API implementation is restarting. Resubmit the request.
CPA_STATUS_UNSUPPORTED	Function is not supported.

Precondition

The component has been initialized via cpaCyStartInstance function.

Postcondition

None

Note

When pCb is non-NULL an asynchronous callback of type CpaCyDsaVerifyCbFunc is generated in response to this function call. For optimal performance, data pointers SHOULD be 8-byte aligned.

See also

CpaCyDsaVerifyOpData, CpaCyDsaVerifyCbFunc

4.12.3.8 cpaCyDsaQueryStats()

```
CpaStatus CPA_DEPRECATED cpaCyDsaQueryStats (
            const CpaInstanceHandle instanceHandle,
            struct _CpaCyDsaStats * pDsaStats )
```

Query statistics for a specific DSA instance.

Deprecated As of v1.3 of the Crypto API, this function has been deprecated, replaced by cpaCyDsaQueryStats64().

Description:

This function will query a specific instance of the DSA implementation for statistics. The user MUST allocate the CpaCyDsaStats structure and pass the reference to that structure into this function call. This function writes the statistic results into the passed in CpaCyDsaStats structure.

Note: statistics returned by this function do not interrupt current data processing and as such can be slightly out of sync with operations that are in progress during the statistics retrieval process.

Context:

This is a synchronous function and it can sleep. It MUST NOT be executed in a context that DOES NOT permit sleeping.

Assumptions:

None

Side-Effects:

None

Blocking:

This function is synchronous and blocking.

Reentrant:

No

Thread-safe:

Yes

Parameters

in	*instanceHandle*	Instance handle.
out	*pDsaStats*	Pointer to memory into which the statistics will be written.

Return values

CPA_STATUS_SUCCESS	Function executed successfully.
CPA_STATUS_FAIL	Function failed.
CPA_STATUS_INVALID_PARAM	Invalid parameter passed in.
CPA_STATUS_RESOURCE	Error related to system resources.
CPA_STATUS_RESTARTING	API implementation is restarting. Resubmit the request.
CPA_STATUS_UNSUPPORTED	Function is not supported.

Precondition

Component has been initialized.

Postcondition

None

Note

This function operates in a synchronous manner and no asynchronous callback will be generated.

See also

CpaCyDsaStats

4.12.3.9 cpaCyDsaQueryStats64()

```
CpaStatus cpaCyDsaQueryStats64 (
            const CpaInstanceHandle instanceHandle,
            CpaCyDsaStats64 * pDsaStats )
```

Query 64-bit statistics for a specific DSA instance.

Description:

This function will query a specific instance of the DSA implementation for 64-bit statistics. The user MUST allocate the CpaCyDsaStats64 structure and pass the reference to that structure into this function. This function writes the statistic results into the passed in CpaCyDsaStats64 structure.

Note: statistics returned by this function do not interrupt current data processing and as such can be slightly out of sync with operations that are in progress during the statistics retrieval process.

Context:

This is a synchronous function and it can sleep. It MUST NOT be executed in a context that DOES NOT permit sleeping.

Assumptions:

None

Side-Effects:

None

Blocking:

This function is synchronous and blocking.

Reentrant:

No

Thread-safe:

Yes

Parameters

in	*instanceHandle*	Instance handle.
out	*pDsaStats*	Pointer to memory into which the statistics will be written.

Return values

CPA_STATUS_SUCCESS	Function executed successfully.
CPA_STATUS_FAIL	Function failed.
CPA_STATUS_INVALID_PARAM	Invalid parameter passed in.
CPA_STATUS_RESOURCE	Error related to system resources.
CPA_STATUS_RESTARTING	API implementation is restarting. Resubmit the request.
CPA_STATUS_UNSUPPORTED	Function is not supported.

Precondition

Component has been initialized.

Postcondition

None

Note

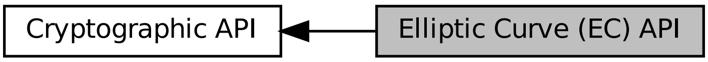
This function operates in a synchronous manner and no asynchronous callback will be generated.

See also

CpaCyDsaStats

4.13 Elliptic Curve (EC) API

Collaboration diagram for Elliptic Curve (EC) API:



Data Structures

- struct _CpaCyEcCurveParametersWeierstrass
- union _CpaCyEcCurveParameters
- struct _CpaCyEcCurve
- struct _CpaCyEcPointMultiplyOpData
- struct _CpaCyEcGenericPointMultiplyOpData
- struct _CpaCyEcGenericPointVerifyOpData
- struct _CpaCyEcMontEdwdsPointMultiplyOpData
- struct _CpaCyEcPointVerifyOpData
- struct _CpaCyEcStats64

Typedefs

- typedef enum _CpaCyEcFieldType CpaCyEcFieldType
- typedef enum _CpaCyEcCurveType CpaCyEcCurveType
- typedef enum _CpaCyEcMontEdwdsCurveType CpaCyEcMontEdwdsCurveType
- typedef struct _CpaCyEcCurveParametersWeierstrass CpaCyEcCurveParametersWeierstrass
- typedef union _CpaCyEcCurveParameters CpaCyEcCurveParameters
- typedef struct _CpaCyEcCurve CpaCyEcCurve
- typedef struct _CpaCyEcPointMultiplyOpData CPA_DEPRECATED
- typedef struct _CpaCyEcGenericPointMultiplyOpData CpaCyEcGenericPointMultiplyOpData
- typedef struct _CpaCyEcGenericPointVerifyOpData CpaCyEcGenericPointVerifyOpData
- typedef struct _CpaCyEcMontEdwdsPointMultiplyOpData CpaCyEcMontEdwdsPointMultiplyOpData
- typedef struct _CpaCyEcStats64 CpaCyEcStats64
- typedef void(∗ CpaCyEcPointMultiplyCbFunc) (void ∗pCallbackTag, CpaStatus status, void ∗pOpData, CpaBoolean multiplyStatus, CpaFlatBuffer ∗pXk, CpaFlatBuffer ∗pYk)
- typedef void(∗ CpaCyEcPointVerifyCbFunc) (void ∗pCallbackTag, CpaStatus status, void ∗pOpData, CpaBoolean verifyStatus)

Enumerations

- enum _CpaCyEcFieldType
- enum _CpaCyEcCurveType
- enum _CpaCyEcMontEdwdsCurveType

Functions

- CpaStatus CPA_DEPRECATED cpaCyEcPointMultiply (const CpaInstanceHandle instanceHandle, const CpaCyEcPointMultiplyCbFunc pCb, void ∗pCallbackTag, const CpaCyEcPointMultiplyOpData ∗pOpData, CpaBoolean ∗pMultiplyStatus, CpaFlatBuffer ∗pXk, CpaFlatBuffer ∗pYk)
- CpaStatus CPA_DEPRECATED cpaCyEcPointVerify (const CpaInstanceHandle instanceHandle, const CpaCyEcPointVerifyCbFunc pCb, void ∗pCallbackTag, const CpaCyEcPointVerifyOpData ∗pOpData, CpaBoolean ∗pVerifyStatus)
- CpaStatus cpaCyEcGenericPointMultiply (const CpaInstanceHandle instanceHandle, const CpaCyEcPointMultiplyCbFunc pCb, void ∗pCallbackTag, const CpaCyEcGenericPointMultiplyOpData ∗pOpData, CpaBoolean ∗pMultiplyStatus, CpaFlatBuffer ∗pXk, CpaFlatBuffer ∗pYk)
- CpaStatus cpaCyEcGenericPointVerify (const CpaInstanceHandle instanceHandle, const CpaCyEcPointVerifyCbFunc pCb, void ∗pCallbackTag, const CpaCyEcGenericPointVerifyOpData ∗pOpData, CpaBoolean ∗pVerifyStatus)
- CpaStatus cpaCyEcMontEdwdsPointMultiply (const CpaInstanceHandle instanceHandle, const CpaCyEcPointMultiplyCbFunc pCb, void ∗pCallbackTag, const CpaCyEcMontEdwdsPointMultiplyOpData ∗pOpData, CpaBoolean ∗pMultiplyStatus, CpaFlatBuffer ∗pXk, CpaFlatBuffer ∗pYk)
- CpaStatus cpaCyEcQueryStats64 (const CpaInstanceHandle instanceHandle, CpaCyEcStats64 ∗pEcStats)

4.13.1 Detailed Description

File: cpa_cy_ec.h

Description:

These functions specify the API for Public Key Encryption (Cryptography) Elliptic Curve (EC) operations.

All implementations will support at least the following:

- "NIST RECOMMENDED ELLIPTIC CURVES FOR FEDERAL GOVERNMENT USE" as defined by `http://csrc.nist.gov/groups/ST/toolkit/documents/dss/NISTReCur.pdf`
- Random curves where the max(log2(q), log2(n) + log2(h)) <= 512 where q is the modulus, n is the order of the curve and h is the cofactor

For Montgomery and Edwards 25519 and 448 elliptic curves, the following operations are supported:

1. Montgomery 25519 Curve | scalar point Multiplication Input: Montgomery affine coordinate X of point P Scalar k Output: Montgomery affine coordinate X of point [k]P Decode: Scalar k always decoded by implementation
2. Montgomery 25519 Curve | generator point Multiplication Input: Scalar k Output: Montgomery affine coordinate X of point [k]G Decode: Scalar k always decoded by implementation
3. Twisted Edwards 25519 Curve | scalar point Multiplication Input: Twisted Edwards affine coordinate X of point P Twisted Edwards affine coordinate Y of point P Scalar k Output: Twisted Edwards affine coordinate X of point [k]P Twisted Edwards affine coordinate Y of point [k]P Decode: Caller must supply parameters in MSB order, the implementation will not explicitly decode according to RFC#7748 Section 5
4. Twisted Edwards 25519 Curve | generator point Multiplication Input: Scalar k Output: Twisted Edwards affine coordinate X of point [k]G Twisted Edwards affine coordinate Y of point [k]G Decode: Caller must supply parameters in MSB order, the implementation will not explicitly decode according to RFC#7748 Section 5

5. Montgomery 448 Curve | scalar point Multiplication Input: Montgomery affine coordinate X of point P Scalar k Output: Montgomery affine coordinate X of point [k]P Decode: Scalar k always decoded by implementation

6. Montgomery 448 Curve | generator point Multiplication Input: Scalar k Output: Montgomery affine coordinate X of point [k]G Decode: Scalar k always decoded by implementation

7. Edwards 448 Curve | scalar point Multiplication Input: Edwards affine coordinate X of point P Edwards affine coordinate Y of point P Scalar k Output: Edwards affine coordinate X of point [k]P Edwards affine coordinate Y of point [k]P Decode: Caller must supply parameters in MSB order, the implementation will not explicitly decode according to RFC#7748 Section 5

8. Edwards 448 Curve | generator point Multiplication Input: Scalar k Output: Edwards affine coordinate X of point [k]G Edwards affine coordinate Y of point [k]G Decode: Caller must supply parameters in MSB order, the implementation will not explicitly decode according to RFC#7748 Section 5

Note

Large numbers are represented on the QuickAssist API as described in the Large Number API (Cryptographic Large Number API).

In addition, the bit length of large numbers passed to the API MUST NOT exceed 576 bits for Elliptic Curve operations.

4.13.2 Typedef Documentation

4.13.2.1 CpaCyEcFieldType

```
typedef enum _CpaCyEcFieldType CpaCyEcFieldType
```

Field types for Elliptic Curve

Description:

As defined by FIPS-186-3, for each cryptovariable length, there are two kinds of fields.

- A prime field is the field GF(p) which contains a prime number p of elements. The elements of this field are the integers modulo p, and the field arithmetic is implemented in terms of the arithmetic of integers modulo p.
- A binary field is the field GF(2^m) which contains 2^m elements for some m (called the degree of the field). The elements of this field are the bit strings of length m, and the field arithmetic is implemented in terms of operations on the bits.

4.13.2.2 CpaCyEcCurveType

```
typedef enum _CpaCyEcCurveType CpaCyEcCurveType
```

Enumeration listing curve types to use with generic multiplication and verification routines.

Description:

This structure contains a list of different elliptic curve types. EC Point multiplication and other operations depend on the type of the curve.

See also

cpaCyEcGenericPointMultiply() cpaCyEcGenericPointVerify()

4.13.2.3 CpaCyEcMontEdwdsCurveType

```
typedef enum _CpaCyEcMontEdwdsCurveType CpaCyEcMontEdwdsCurveType
```

Curve types for Elliptic Curves defined in RFC#7748

Description:

As defined by RFC 7748, there are four elliptic curves in this group. The Montgomery curves are denoted curve25519 and curve448, and the birationally equivalent Twisted Edwards curves are denoted edwards25519 and edwards448

4.13.2.4 CpaCyEcCurveParametersWeierstrass

```
typedef struct _CpaCyEcCurveParametersWeierstrass CpaCyEcCurveParametersWeierstrass
```

Curve parameters for a Weierstrass type curve.

Description:

This structure contains curve parameters for Weierstrass type curve: y^2 = x^3 + ax + b The client MUST allocate the memory for this structure When the structure is passed into the function, ownership of the memory passes to the function. Ownership of the memory returns to the client when this structure is returned in the callback function.

For optimal performance all data buffers SHOULD be 8-byte aligned. The legend used in this structure is borrowed from RFC7748

Note

If the client modifies or frees the memory referenced in this structure after it has been submitted to the function, and before it has been returned in the callback, undefined behavior will result.

See also

CpaCyEcCurveParameters CpaCyEcFieldType

4.13.2.5 CpaCyEcCurveParameters

```
typedef union _CpaCyEcCurveParameters CpaCyEcCurveParameters
```

Union characterised by a specific curve.

Description:

This union allows for the characterisation of different curve types encapsulted in one data type. The intention is that new curve types will be added in the future.

Note

See also

CpaCyEcCurveParametersWeierstrass

4.13.2.6 CpaCyEcCurve

```
typedef struct _CpaCyEcCurve CpaCyEcCurve
```

Unified curve parameters.

Description:

This structure provides a single data type that can describe a number of different curve types. The intention is to add further curve types in the future, thus the union field will allow for that expansion.

The client MUST allocate the memory for this structure and the items pointed to by this structure. When the structure is passed into the function, ownership of the memory passes to the function. Ownership of the memory returns to the client when this structure is returned in the callback function.

For optimal performance all data buffers SHOULD be 8-byte aligned.

Note

If the client modifies or frees the memory referenced in this structure after it has been submitted to the function, and before it has been returned in the callback, undefined behavior will result.

See also

CpaCyEcCurveParameters cpaCyEcGenericPointMultiply cpaCyEcGenericPointVerify

4.13.2.7 CPA_DEPRECATED

```
typedef struct _CpaCyEcPointVerifyOpData CPA_DEPRECATED
```

EC Point Multiplication Operation Data.

Description:

This structure contains the operation data for the cpaCyEcPointMultiply function. The client MUST allocate the memory for this structure and the items pointed to by this structure. When the structure is passed into the function, ownership of the memory passes to the function. Ownership of the memory returns to the client when this structure is returned in the callback function.

For optimal performance all data buffers SHOULD be 8-byte aligned.

All values in this structure are required to be in Most Significant Byte first order, e.g. a.pData[0] = MSB.

Note

If the client modifies or frees the memory referenced in this structure after it has been submitted to the cpaCyEcPointMultiply function, and before it has been returned in the callback, undefined behavior will result.

See also

cpaCyEcPointMultiply()

EC Point Verification Operation Data.

Description:

This structure contains the operation data for the cpaCyEcPointVerify function. The client MUST allocate the memory for this structure and the items pointed to by this structure. When the structure is passed into the function, ownership of the memory passes to the function. Ownership of the memory returns to the client when this structure is returned in the callback function.

For optimal performance all data buffers SHOULD be 8-byte aligned.

All values in this structure are required to be in Most Significant Byte first order, e.g. a.pData[0] = MSB.

Note

If the client modifies or frees the memory referenced in this structure after it has been submitted to the CpaCyEcPointVerify function, and before it has been returned in the callback, undefined behavior will result.

See also

cpaCyEcPointVerify()

4.13.2.8 CpaCyEcGenericPointMultiplyOpData

```
typedef struct _CpaCyEcGenericPointMultiplyOpData CpaCyEcGenericPointMultiplyOpData
```

Generic EC Point Multiplication Operation Data.

Description:

This structure contains a generic EC point and a multiplier for use with cpaCyEcGenericPointMultiply. This is common for representing all EC points, irrespective of curve type: Weierstrass, Montgomery and Twisted Edwards (at this time only Weierstrass are supported). The same point + multiplier format can be used when performing generator multiplication, in which case the xP, yP supplied in this structure will be ignored by QAT API library & a generator point will be inserted in their place.

For optimal performance all data buffers SHOULD be 8-byte aligned.

All values in this structure are required to be in Most Significant Byte first order, e.g. a.pData[0] = MSB.

Note

If the client modifies or frees the memory referenced in this structure after it has been submitted to the cpaCyEcGenericPointMultiply function, and before it has been returned in the callback, undefined behavior will result.

See also

cpaCyEcGenericPointMultiply()

4.13.2.9 CpaCyEcGenericPointVerifyOpData

```
typedef struct _CpaCyEcGenericPointVerifyOpData CpaCyEcGenericPointVerifyOpData
```

Generic EC Point Verify Operation Data.

Description:

This structure contains the operation data for the cpaCyEcGenericPointVerify function. This is common for representing all EC points, irrespective of curve type: Weierstrass, Montgomery and Twisted Edwards (at this time only Weierstrass are supported).

This structure contains a generic EC point, irrespective of curve type. It is used to verify when the <x,y> pair specified in the structure lies on the curve indicated in the cpaCyEcGenericPointVerify API.

For optimal performance all data buffers SHOULD be 8-byte aligned.

All values in this structure are required to be in Most Significant Byte first order, e.g. a.pData[0] = MSB.

Note

If the client modifies or frees the memory referenced in this structure after it has been submitted to the cpaCyEcGenericPointVerify function, and before it has been returned in the callback, undefined behavior will result.

See also

cpaCyEcGenericPointVerify()

4.13.2.10 CpaCyEcMontEdwdsPointMultiplyOpData

```
typedef struct _CpaCyEcMontEdwdsPointMultiplyOpData CpaCyEcMontEdwdsPointMultiplyOpData
```

EC Point Multiplication Operation Data for Edwards or Montgomery curves as specificied in RFC#7748.

Description:

This structure contains the operation data for the cpaCyEcMontEdwdsPointMultiply function. The client MUST allocate the memory for this structure and the items pointed to by this structure. When the structure is passed into the function, ownership of the memory passes to the function. Ownership of the memory returns to the client when this structure is returned in the callback function.

For optimal performance all data buffers SHOULD be 8-byte aligned.

All values in this structure are required to be in Most Significant Byte first order, e.g. a.pData[0] = MSB.

Note

If the client modifies or frees the memory referenced in this structure after it has been submitted to the cpaCyEcMontEdwdsPointMultiply function, and before it has been returned in the callback, undefined behavior will result.

All buffers in this structure need to be:

- 32 bytes in size for 25519 curves
- 64 bytes in size for 448 curves

See also

cpaCyEcMontEdwdsPointMultiply()

4.13.2.11 CpaCyEcStats64

```
typedef struct _CpaCyEcStats64 CpaCyEcStats64
```

Cryptographic EC Statistics.

Description:

This structure contains statistics on the Cryptographic EC operations. Statistics are set to zero when the component is initialized, and are collected per instance.

4.13.2.12 CpaCyEcPointMultiplyCbFunc

```
typedef void(* CpaCyEcPointMultiplyCbFunc) (void *pCallbackTag, CpaStatus status, void
*pOpData, CpaBoolean multiplyStatus, CpaFlatBuffer *pXk, CpaFlatBuffer *pYk)
```

Definition of callback function invoked for cpaCyEcPointMultiply requests.

Context:

This callback function can be executed in a context that DOES NOT permit sleeping to occur.

Assumptions:

None

Side-Effects:

None

Reentrant:

No

Thread-safe:

Yes

Parameters

in	*pCallbackTag*	User-supplied value to help identify request.
in	*status*	Status of the operation. Valid values are CPA_STATUS_SUCCESS, CPA_STATUS_FAIL and CPA_STATUS_UNSUPPORTED.
in	*pOpData*	Opaque pointer to Operation data supplied in request.
in	*multiplyStatus*	Status of the point multiplication.
in	*pXk*	x coordinate of resultant EC point.
in	*pYk*	y coordinate of resultant EC point.

Return values

None	

Precondition

Component has been initialized.

Postcondition

None

Note

None

See also

cpaCyEcGenericPointMultiply()

Definition at line 601 of file cpa_cy_ec.h.

4.13.2.13 CpaCyEcPointVerifyCbFunc

```
typedef void(* CpaCyEcPointVerifyCbFunc) (void *pCallbackTag, CpaStatus status, void *pOpData,
CpaBoolean verifyStatus)
```

Definition of callback function invoked for cpaCyEcGenericPointVerify requests.

Context:

This callback function can be executed in a context that DOES NOT permit sleeping to occur.

Assumptions:

None

Side-Effects:

None

Reentrant:

No

Thread-safe:

Yes

Parameters

in	*pCallbackTag*	User-supplied value to help identify request.
in	*status*	Status of the operation. Valid values are CPA_STATUS_SUCCESS, CPA_STATUS_FAIL and CPA_STATUS_UNSUPPORTED.
in	*pOpData*	Operation data pointer supplied in request.
in	*verifyStatus*	Set to CPA_FALSE if the point is NOT on the curve or at infinity. Set to CPA_TRUE if the point is on the curve.

Returns

None

Precondition

Component has been initialized.

Postcondition

None

Note

None

See also

cpaCyEcGenericPointVerify()

Definition at line 647 of file cpa_cy_ec.h.

4.13.3 Enumeration Type Documentation

4.13.3.1 _CpaCyEcFieldType

```
enum _CpaCyEcFieldType
```

Field types for Elliptic Curve

Description:

As defined by FIPS-186-3, for each cryptovariable length, there are two kinds of fields.

- A prime field is the field GF(p) which contains a prime number p of elements. The elements of this field are the integers modulo p, and the field arithmetic is implemented in terms of the arithmetic of integers modulo p.
- A binary field is the field GF(2^m) which contains 2^m elements for some m (called the degree of the field). The elements of this field are the bit strings of length m, and the field arithmetic is implemented in terms of operations on the bits.

Enumerator

CPA_CY_EC_FIELD_TYPE_PRIME	A prime field, GF(p)
CPA_CY_EC_FIELD_TYPE_BINARY	A binary field, GF(2^m)

Definition at line 131 of file cpa_cy_ec.h.

4.13.3.2 _CpaCyEcCurveType

```
enum _CpaCyEcCurveType
```

Enumeration listing curve types to use with generic multiplication and verification routines.

Description:

This structure contains a list of different elliptic curve types. EC Point multiplication and other operations depend on the type of the curve.

See also

cpaCyEcGenericPointMultiply() cpaCyEcGenericPointVerify()

Enumerator

CPA_CY_EC_CURVE_TYPE_WEIERSTRASS_PRIME	A Weierstrass curve with arithmetic in terms of the arithmetic of integers modulo p over a prime field.
CPA_CY_EC_CURVE_TYPE_WEIERSTRASS_BINARY	A Weierstrass curve with arithmetic in terms of operations on bits over a binary field.
CPA_CY_EC_CURVE_TYPE_WEIERSTRASS_KOBLITZ_BINARY	A Weierstrass-koblitz curve with arithmetic in terms of operations on the bits over a binary field.

Definition at line 155 of file cpa_cy_ec.h.

4.13.3.3 _CpaCyEcMontEdwdsCurveType

```
enum _CpaCyEcMontEdwdsCurveType
```

Curve types for Elliptic Curves defined in RFC#7748

Description:

As defined by RFC 7748, there are four elliptic curves in this group. The Montgomery curves are denoted curve25519 and curve448, and the birationally equivalent Twisted Edwards curves are denoted edwards25519 and edwards448

Enumerator

CPA_CY_EC_MONTEDWDS_CURVE25519_TYPE	Montgomery 25519 curve
CPA_CY_EC_MONTEDWDS_ED25519_TYPE	Edwards 25519 curve
CPA_CY_EC_MONTEDWDS_CURVE448_TYPE	Montgomery 448 curve
CPA_CY_EC_MONTEDWDS_ED448_TYPE	Edwards 448 curve

Definition at line 180 of file cpa_cy_ec.h.

4.13.4 Function Documentation

4.13.4.1 cpaCyEcPointMultiply()

```
CpaStatus CPA_DEPRECATED cpaCyEcPointMultiply (
            const CpaInstanceHandle instanceHandle,
            const CpaCyEcPointMultiplyCbFunc pCb,
            void * pCallbackTag,
            const CpaCyEcPointMultiplyOpData * pOpData,
            CpaBoolean * pMultiplyStatus,
            CpaFlatBuffer * pXk,
            CpaFlatBuffer * pYk )
```

Perform EC Point Multiplication.

Deprecated This function is replaced with cpaCyEcGenericPointMultiply

Description:

This function performs Elliptic Curve Point Multiplication as per ANSI X9.63 Annex D.3.2.

Context:

When called as an asynchronous function it cannot sleep. It can be executed in a context that does not permit sleeping. When called as a synchronous function it may sleep. It MUST NOT be executed in a context that DOES NOT permit sleeping.

Assumptions:

None

Side-Effects:

None

Blocking:

Yes when configured to operate in synchronous mode.

Reentrant:

No

Thread-safe:

Yes

Parameters

in	*instanceHandle*	Instance handle.
in	*pCb*	Callback function pointer. If this is set to a NULL value the function will operate synchronously.
in	*pCallbackTag*	User-supplied value to help identify request.
in	*pOpData*	Structure containing all the data needed to perform the operation. The client code allocates the memory for this structure. This component takes ownership of the memory until it is returned in the callback.
out	*pMultiplyStatus*	In synchronous mode, the multiply output is valid (CPA_TRUE) or the output is invalid (CPA_FALSE).
out	*pXk*	Pointer to xk flat buffer.
out	*pYk*	Pointer to yk flat buffer.

Return values

CPA_STATUS_SUCCESS	Function executed successfully.
CPA_STATUS_FAIL	Function failed.
CPA_STATUS_RETRY	Resubmit the request.
CPA_STATUS_INVALID_PARAM	Invalid parameter in.
CPA_STATUS_RESOURCE	Error related to system resources.
CPA_STATUS_RESTARTING	API implementation is restarting. Resubmit the request.
CPA_STATUS_UNSUPPORTED	Function is not supported.

Precondition

The component has been initialized via cpaCyStartInstance function.

Postcondition

None

Note

When pCb is non-NULL an asynchronous callback of type CpaCyEcPointMultiplyCbFunc is generated in response to this function call. For optimal performance, data pointers SHOULD be 8-byte aligned.

See also

CpaCyEcPointMultiplyOpData, CpaCyEcPointMultiplyCbFunc

4.13.4.2 cpaCyEcPointVerify()

```
CpaStatus CPA_DEPRECATED cpaCyEcPointVerify (
            const CpaInstanceHandle instanceHandle,
            const CpaCyEcPointVerifyCbFunc pCb,
            void * pCallbackTag,
            const CpaCyEcPointVerifyOpData * pOpData,
            CpaBoolean * pVerifyStatus )
```

Verify that a point is on an elliptic curve.

Deprecated This function is replaced with cpaCyEcGenericPointVerify

Description:

This function performs Elliptic Curve Point Verification, as per steps a, b and c of ANSI X9.62 Annex A.4.2. (To perform the final step d, the user can call cpaCyEcPointMultiply.)

This function checks if the specified point satisfies the Weierstrass equation for an Elliptic Curve.

For GF(p): y^2 = (x^3 + ax + b) mod p For GF(2^m): y^2 + xy = x^3 + ax^2 + b mod p where p is the irreducible polynomial over GF(2^m)

Use this function to verify a point is in the correct range and is NOT the point at infinity.

Context:

When called as an asynchronous function it cannot sleep. It can be executed in a context that does not permit sleeping. When called as a synchronous function it may sleep. It MUST NOT be executed in a context that DOES NOT permit sleeping.

Assumptions:

None

Side-Effects:

None

Blocking:

Yes when configured to operate in synchronous mode.

Reentrant:

No

Thread-safe:

Yes

Parameters

in	*instanceHandle*	Instance handle.
in	*pCb*	Callback function pointer. If this is set to a NULL value the function will operate synchronously.
in	*pCallbackTag*	User-supplied value to help identify request.
in	*pOpData*	Structure containing all the data needed to perform the operation. The client code allocates the memory for this structure. This component takes ownership of the memory until it is returned in the callback.
out	*pVerifyStatus*	In synchronous mode, set to CPA_FALSE if the point is NOT on the curve or at infinity. Set to CPA_TRUE if the point is on the curve.

Return values

CPA_STATUS_SUCCESS	Function executed successfully.
CPA_STATUS_FAIL	Function failed.
CPA_STATUS_RETRY	Resubmit the request.
CPA_STATUS_INVALID_PARAM	Invalid parameter passed in.
CPA_STATUS_RESOURCE	Error related to system resources.
CPA_STATUS_RESTARTING	API implementation is restarting. Resubmit the request.
CPA_STATUS_UNSUPPORTED	Function is not supported.

Precondition

The component has been initialized via cpaCyStartInstance function.

Postcondition

None

Note

When pCb is non-NULL an asynchronous callback of type CpaCyEcPointVerifyCbFunc is generated in response to this function call. For optimal performance, data pointers SHOULD be 8-byte aligned.

See also

CpaCyEcPointVerifyOpData, CpaCyEcPointVerifyCbFunc

4.13.4.3 cpaCyEcGenericPointMultiply()

```
CpaStatus cpaCyEcGenericPointMultiply (
            const CpaInstanceHandle instanceHandle,
            const CpaCyEcPointMultiplyCbFunc pCb,
            void * pCallbackTag,
            const CpaCyEcGenericPointMultiplyOpData * pOpData,
            CpaBoolean * pMultiplyStatus,
            CpaFlatBuffer * pXk,
            CpaFlatBuffer * pYk )
```

Generic ECC point multiplication operation.

Description:

This is the generic ECC point multiplication operation, which is agnostic to the type of the curve used.

Context:

Assumptions:

None

Side-Effects:

None

Blocking:

Yes when configured to operate in synchronous mode.

Reentrant:

No

Thread-safe:

Yes

Parameters

in	*instanceHandle*	Instance handle.
in	*pCb*	Callback function pointer. If this is set to a NULL value, the function will operate synchronously.
in	*pCallbackTag*	User-supplied value to help identify request.
in	*pOpData*	Structure containing all the data needed to perform the operation. The client code allocates the memory for this structure. This component takes ownership of the memory until it is returned in the callback.
out	*pMultiplyStatus*	In synchronous mode, the multiply output is valid (CPA_TRUE) or the output is invalid (CPA_FALSE).
out	*pXk*	Pointer to xk flat buffer.
out	*pYk*	Pointer to yk flat buffer.

Return values

CPA_STATUS_SUCCESS	Function executed successfully.
CPA_STATUS_FAIL	Function failed.
CPA_STATUS_INVALID_PARAM	Invalid parameter passed in.
CPA_STATUS_RESOURCE	Error related to system resources.
CPA_STATUS_RESTARTING	API implementation is restarting. Resubmit the request.
CPA_STATUS_UNSUPPORTED	Function is not supported.

Precondition

Component has been initialized.

Postcondition

None

Note

When pCb is non-NULL an asynchronous callback of type CpaCyEcPointMultiplyCbFunc is generated in response to this function call. For optimal performance, data pointers SHOULD be 8-byte aligned.

See also

CpaCyEcPointMultiplyOpData, CpaCyEcPointMultiplyCbFunc CpaCyEcCurveType CpaCyEcCurveParameters

4.13.4.4 cpaCyEcGenericPointVerify()

```
CpaStatus cpaCyEcGenericPointVerify (
            const CpaInstanceHandle instanceHandle,
            const CpaCyEcPointVerifyCbFunc pCb,
            void * pCallbackTag,
            const CpaCyEcGenericPointVerifyOpData * pOpData,
            CpaBoolean * pVerifyStatus )
```

Generic ECC point verification operation.

Description:

This is the generic ECC point verification operation, which is agnostic to the type of the curve used.

Context:

Assumptions:

None

Side-Effects:

None

Blocking:

Yes when configured to operate in synchronous mode.

Reentrant:

No

Thread-safe:

Yes

Parameters

in	*instanceHandle*	Instance handle.
in	*pCb*	Callback function pointer. If this is set to a NULL value the function will operate synchronously.
in	*pCallbackTag*	User-supplied value to help identify request.
in	*pOpData*	Structure containing all the data needed to perform the operation. The client code allocates the memory for this structure. This component takes ownership of the memory until it is returned in the callback.
out	*pVerifyStatus*	In synchronous mode, the verification output is valid (CPA_TRUE) or the output is invalid (CPA_FALSE).

Return values

CPA_STATUS_SUCCESS	Function executed successfully.
CPA_STATUS_FAIL	Function failed.
CPA_STATUS_INVALID_PARAM	Invalid parameter passed in.
CPA_STATUS_RESOURCE	Error related to system resources.
CPA_STATUS_RESTARTING	API implementation is restarting. Resubmit the request.
CPA_STATUS_UNSUPPORTED	Function is not supported.

Precondition

Component has been initialized.

Postcondition

None

Note

When pCb is non-NULL an asynchronous callback of type CpaCyEcPointVerifyCbFunc is generated in response to this function call. For optimal performance, data pointers SHOULD be 8-byte aligned.

See also

CpaCyEcGenericPointVerifyOpData, CpaCyEcPointVerifyCbFunc CpaCyEcCurveType CpaCyEcCurveParameters

4.13.4.5 cpaCyEcMontEdwdsPointMultiply()

```
CpaStatus cpaCyEcMontEdwdsPointMultiply (
        const CpaInstanceHandle instanceHandle,
        const CpaCyEcPointMultiplyCbFunc pCb,
        void * pCallbackTag,
        const CpaCyEcMontEdwdsPointMultiplyOpData * pOpData,
        CpaBoolean * pMultiplyStatus,
        CpaFlatBuffer * pXk,
        CpaFlatBuffer * pYk )
```

Perform EC Point Multiplication on an Edwards or Montgomery curve as defined in RFC#7748.

Description:

This function performs Elliptic Curve Point Multiplication as per RFC#7748

Context:

When called as an asynchronous function it cannot sleep. It can be executed in a context that does not permit sleeping. When called as a synchronous function it may sleep. It MUST NOT be executed in a context that DOES NOT permit sleeping.

Assumptions:

None

Side-Effects:

None

Blocking:

Yes when configured to operate in synchronous mode.

Reentrant:

No

Thread-safe:

Yes

Parameters

in	*instanceHandle*	Instance handle.
in	*pCb*	Callback function pointer. If this is set to a NULL value the function will operate synchronously.
in	*pCallbackTag*	User-supplied value to help identify request.
in	*pOpData*	Structure containing all the data needed to perform the operation. The client code allocates the memory for this structure. This component takes ownership of the memory until it is returned in the callback.
out	*pMultiplyStatus*	In synchronous mode, the multiply output is valid (CPA_TRUE) or the output is invalid (CPA_FALSE).
out	*pXk*	Pointer to xk flat buffer.
out	*pYk*	Pointer to yk flat buffer.

Return values

CPA_STATUS_SUCCESS	Function executed successfully.
CPA_STATUS_FAIL	Function failed.
CPA_STATUS_RETRY	Resubmit the request.
CPA_STATUS_INVALID_PARAM	Invalid parameter in.

Return values

CPA_STATUS_RESOURCE	Error related to system resources.
CPA_STATUS_RESTARTING	API implementation is restarting. Resubmit the request.
CPA_STATUS_UNSUPPORTED	Function is not supported.

Precondition

The component has been initialized via cpaCyStartInstance function.

Postcondition

None

Note

When pCb is non-NULL an asynchronous callback of type CpaCyEcPointMultiplyCbFunc is generated in response to this function call. For optimal performance, data pointers SHOULD be 8-byte aligned.

See also

CpaCyEcMontEdwdsPointMultiplyOpData, CpaCyEcMontEdwdsPointMultiplyCbFunc

4.13.4.6 cpaCyEcQueryStats64()

```
CpaStatus cpaCyEcQueryStats64 (
            const CpaInstanceHandle instanceHandle,
            CpaCyEcStats64 * pEcStats )
```

Query statistics for a specific EC instance.

Description:

This function will query a specific instance of the EC implementation for statistics. The user MUST allocate the CpaCyEcStats64 structure and pass the reference to that structure into this function call. This function writes the statistic results into the passed in CpaCyEcStats64 structure.

Note: statistics returned by this function do not interrupt current data processing and as such can be slightly out of sync with operations that are in progress during the statistics retrieval process.

Context:

This is a synchronous function and it can sleep. It MUST NOT be executed in a context that DOES NOT permit sleeping.

Assumptions:

None

Side-Effects:

None

Blocking:

This function is synchronous and blocking.

Reentrant:

No

Thread-safe:

Yes

Parameters

in	*instanceHandle*	Instance handle.
out	*pEcStats*	Pointer to memory into which the statistics will be written.

Return values

CPA_STATUS_SUCCESS	Function executed successfully.
CPA_STATUS_FAIL	Function failed.
CPA_STATUS_INVALID_PARAM	Invalid parameter passed in.
CPA_STATUS_RESOURCE	Error related to system resources.
CPA_STATUS_RESTARTING	API implementation is restarting. Resubmit the request.
CPA_STATUS_UNSUPPORTED	Function is not supported.

Precondition

Component has been initialized.

Postcondition

None

Note

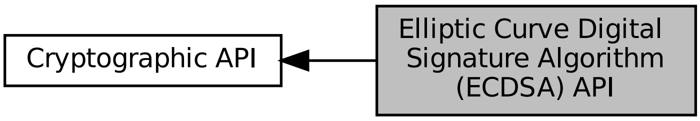
This function operates in a synchronous manner and no asynchronous callback will be generated.

See also

CpaCyEcStats64

4.14 Elliptic Curve Digital Signature Algorithm (ECDSA) API

Collaboration diagram for Elliptic Curve Digital Signature Algorithm (ECDSA) API:



Data Structures

- struct _CpaCyEcdsaSignROpData
- struct _CpaCyEcdsaSignSOpData
- struct _CpaCyEcdsaSignRSOpData
- struct _CpaCyEcdsaVerifyOpData
- struct _CpaCyEcdsaStats64

Typedefs

- typedef struct _CpaCyEcdsaSignROpData CpaCyEcdsaSignROpData
- typedef struct _CpaCyEcdsaSignSOpData CpaCyEcdsaSignSOpData
- typedef struct _CpaCyEcdsaSignRSOpData CpaCyEcdsaSignRSOpData
- typedef struct _CpaCyEcdsaVerifyOpData CpaCyEcdsaVerifyOpData
- typedef struct _CpaCyEcdsaStats64 CpaCyEcdsaStats64
- typedef void(∗ CpaCyEcdsaGenSignCbFunc) (void ∗pCallbackTag, CpaStatus status, void ∗pOpData, CpaBoolean multiplyStatus, CpaFlatBuffer ∗pOut)
- typedef void(∗ CpaCyEcdsaSignRSCbFunc) (void ∗pCallbackTag, CpaStatus status, void ∗pOpData, CpaBoolean multiplyStatus, CpaFlatBuffer ∗pR, CpaFlatBuffer ∗pS)
- typedef void(∗ CpaCyEcdsaVerifyCbFunc) (void ∗pCallbackTag, CpaStatus status, void ∗pOpData, CpaBoolean verifyStatus)

Functions

- CpaStatus cpaCyEcdsaSignR (const CpaInstanceHandle instanceHandle, const CpaCyEcdsaGenSignCbFunc pCb, void ∗pCallbackTag, const CpaCyEcdsaSignROpData ∗pOpData, CpaBoolean ∗pSignStatus, CpaFlatBuffer ∗pR)
- CpaStatus cpaCyEcdsaSignS (const CpaInstanceHandle instanceHandle, const CpaCyEcdsaGenSignCbFunc pCb, void ∗pCallbackTag, const CpaCyEcdsaSignSOpData ∗pOpData, CpaBoolean ∗pSignStatus, CpaFlatBuffer ∗pS)
- CpaStatus cpaCyEcdsaSignRS (const CpaInstanceHandle instanceHandle, const CpaCyEcdsaSignRSCbFunc pCb, void ∗pCallbackTag, const CpaCyEcdsaSignRSOpData ∗pOpData, CpaBoolean ∗pSignStatus, CpaFlatBuffer ∗pR, CpaFlatBuffer ∗pS)
- CpaStatus cpaCyEcdsaVerify (const CpaInstanceHandle instanceHandle, const CpaCyEcdsaVerifyCbFunc pCb, void ∗pCallbackTag, const CpaCyEcdsaVerifyOpData ∗pOpData, CpaBoolean ∗pVerifyStatus)
- CpaStatus cpaCyEcdsaQueryStats64 (const CpaInstanceHandle instanceHandle, CpaCyEcdsaStats64 ∗pEcdsaStats)

4.14.1 Detailed Description

File: cpa_cy_ecdsa.h

Description:

These functions specify the API for Public Key Encryption (Cryptography) Elliptic Curve Digital Signature Algorithm (ECDSA) operations.

Note

Large numbers are represented on the QuickAssist API as described in the Large Number API (Cryptographic Large Number API).

In addition, the bit length of large numbers passed to the API MUST NOT exceed 576 bits for Elliptic Curve operations.

4.14.2 Typedef Documentation

4.14.2.1 CpaCyEcdsaSignROpData

```
typedef struct _CpaCyEcdsaSignROpData CpaCyEcdsaSignROpData
```

ECDSA Sign R Operation Data.

Description:

This structure contains the operation data for the cpaCyEcdsaSignR function. The client MUST allocate the memory for this structure and the items pointed to by this structure. When the structure is passed into the function, ownership of the memory passes to the function. Ownership of the memory returns to the client when this structure is returned in the callback function.

For optimal performance all data buffers SHOULD be 8-byte aligned.

All values in this structure are required to be in Most Significant Byte first order, e.g. a.pData[0] = MSB.

Note

If the client modifies or frees the memory referenced in this structure after it has been submitted to the cpaCyEcdsaSignR function, and before it has been returned in the callback, undefined behavior will result.

See also

cpaCyEcdsaSignR()

4.14.2.2 CpaCyEcdsaSignSOpData

```
typedef struct _CpaCyEcdsaSignSOpData CpaCyEcdsaSignSOpData
```

ECDSA Sign S Operation Data.

Description:

This structure contains the operation data for the cpaCyEcdsaSignS function. The client MUST allocate the memory for this structure and the items pointed to by this structure. When the structure is passed into the function, ownership of the memory passes to the function. Ownership of the memory returns to the client when this structure is returned in the callback function.

For optimal performance all data buffers SHOULD be 8-byte aligned.

All values in this structure are required to be in Most Significant Byte first order, e.g. a.pData[0] = MSB.

Note

If the client modifies or frees the memory referenced in this structure after it has been submitted to the cpaCyEcdsaSignS function, and before it has been returned in the callback, undefined behavior will result.

See also

cpaCyEcdsaSignS()

4.14.2.3 CpaCyEcdsaSignRSOpData

```
typedef struct _CpaCyEcdsaSignRSOpData CpaCyEcdsaSignRSOpData
```

ECDSA Sign R & S Operation Data.

Description:

This structure contains the operation data for the cpaCyEcdsaSignRS function. The client MUST allocate the memory for this structure and the items pointed to by this structure. When the structure is passed into the function, ownership of the memory passes to the function. Ownership of the memory returns to the client when this structure is returned in the callback function.

For optimal performance all data buffers SHOULD be 8-byte aligned.

All values in this structure are required to be in Most Significant Byte first order, e.g. a.pData[0] = MSB.

Note

If the client modifies or frees the memory referenced in this structure after it has been submitted to the cpaCyEcdsaSignRS function, and before it has been returned in the callback, undefined behavior will result.

See also

cpaCyEcdsaSignRS()

4.14.2.4 CpaCyEcdsaVerifyOpData

```
typedef struct _CpaCyEcdsaVerifyOpData CpaCyEcdsaVerifyOpData
```

ECDSA Verify Operation Data, for Public Key.

Description:

This structure contains the operation data for the CpaCyEcdsaVerify function. The client MUST allocate the memory for this structure and the items pointed to by this structure. When the structure is passed into the function, ownership of the memory passes to the function. Ownership of the memory returns to the client when this structure is returned in the callback function.

For optimal performance all data buffers SHOULD be 8-byte aligned.

All values in this structure are required to be in Most Significant Byte first order, e.g. a.pData[0] = MSB.

Note

If the client modifies or frees the memory referenced in this structure after it has been submitted to the cpaCyEcdsaVerify function, and before it has been returned in the callback, undefined behavior will result.

See also

CpaCyEcdsaVerify()

4.14.2.5 CpaCyEcdsaStats64

```
typedef struct _CpaCyEcdsaStats64 CpaCyEcdsaStats64
```

Cryptographic ECDSA Statistics.

Description:

This structure contains statistics on the Cryptographic ECDSA operations. Statistics are set to zero when the component is initialized, and are collected per instance.

4.14.2.6 CpaCyEcdsaGenSignCbFunc

```
typedef void(* CpaCyEcdsaGenSignCbFunc) (void *pCallbackTag, CpaStatus status, void *pOpData,
CpaBoolean multiplyStatus, CpaFlatBuffer *pOut)
```

Definition of a generic callback function invoked for a number of the ECDSA Sign API functions.

Description:

This is the prototype for the CpaCyEcdsaGenSignCbFunc callback function.

Context:

This callback function can be executed in a context that DOES NOT permit sleeping to occur.

Assumptions:

None

Side-Effects:

None

Reentrant:

No

Thread-safe:

Yes

Parameters

in	*pCallbackTag*	User-supplied value to help identify request.
in	*status*	Status of the operation. Valid values are CPA_STATUS_SUCCESS, CPA_STATUS_FAIL and CPA_STATUS_UNSUPPORTED.
in	*pOpData*	Opaque pointer to Operation data supplied in request.
in	*multiplyStatus*	Status of the point multiplication.
in	*pOut*	Output data from the request.

Return values

None	

Precondition

Component has been initialized.

Postcondition

None

Note

None

See also

cpaCyEcdsaSignR() cpaCyEcdsaSignS()

Definition at line 372 of file cpa_cy_ecdsa.h.

4.14.2.7 CpaCyEcdsaSignRSCbFunc

```
typedef void(* CpaCyEcdsaSignRSCbFunc) (void *pCallbackTag, CpaStatus status, void *pOpData,
CpaBoolean multiplyStatus, CpaFlatBuffer *pR, CpaFlatBuffer *pS)
```

Definition of callback function invoked for cpaCyEcdsaSignRS requests.

Description:

This is the prototype for the CpaCyEcdsaSignRSCbFunc callback function, which will provide the ECDSA message signature r and s parameters.

Context:

This callback function can be executed in a context that DOES NOT permit sleeping to occur.

Assumptions:

None

Side-Effects:

None

Reentrant:

No

Thread-safe:

Yes

Parameters

in	*pCallbackTag*	User-supplied value to help identify request.
in	*status*	Status of the operation. Valid values are CPA_STATUS_SUCCESS, CPA_STATUS_FAIL and CPA_STATUS_UNSUPPORTED.
in	*pOpData*	Operation data pointer supplied in request.
in	*multiplyStatus*	Status of the point multiplication.
in	*pR*	Ecdsa message signature r.
in	*pS*	Ecdsa message signature s.

Return values

None	

Precondition

Component has been initialized.

Postcondition

None

Note

None

See also

cpaCyEcdsaSignRS()

Definition at line 423 of file cpa_cy_ecdsa.h.

4.14.2.8 CpaCyEcdsaVerifyCbFunc

```
typedef void(* CpaCyEcdsaVerifyCbFunc) (void *pCallbackTag, CpaStatus status, void *pOpData,
CpaBoolean verifyStatus)
```

Definition of callback function invoked for cpaCyEcdsaVerify requests.

Description:

This is the prototype for the CpaCyEcdsaVerifyCbFunc callback function.

Context:

This callback function can be executed in a context that DOES NOT permit sleeping to occur.

Assumptions:

None

Side-Effects:

None

Reentrant:

No

Thread-safe:

Yes

Parameters

in	*pCallbackTag*	User-supplied value to help identify request.
in	*status*	Status of the operation. Valid values are CPA_STATUS_SUCCESS, CPA_STATUS_FAIL and CPA_STATUS_UNSUPPORTED.
in	*pOpData*	Operation data pointer supplied in request.
in	*verifyStatus*	The verification status.

Return values

None	

Precondition

Component has been initialized.

Postcondition

None

Note

None

See also

cpaCyEcdsaVerify()

Definition at line 470 of file cpa_cy_ecdsa.h.

4.14.3 Function Documentation

4.14.3.1 cpaCyEcdsaSignR()

```
CpaStatus cpaCyEcdsaSignR (
            const CpaInstanceHandle instanceHandle,
            const CpaCyEcdsaGenSignCbFunc pCb,
            void * pCallbackTag,
            const CpaCyEcdsaSignROpData * pOpData,
            CpaBoolean * pSignStatus,
            CpaFlatBuffer * pR )
```

Generate ECDSA Signature R.

Description:

This function generates ECDSA Signature R as per ANSI X9.62 2005 section 7.3.

Context:

When called as an asynchronous function it cannot sleep. It can be executed in a context that does not permit sleeping. When called as a synchronous function it may sleep. It MUST NOT be executed in a context that DOES NOT permit sleeping.

Assumptions:

None

Side-Effects:

None

Blocking:

Yes when configured to operate in synchronous mode.

Reentrant:

No

Thread-safe:

Yes

Parameters

in	*instanceHandle*	Instance handle.
in	*pCb*	Callback function pointer. If this is set to a NULL value the function will operate synchronously.
in	*pCallbackTag*	User-supplied value to help identify request.
in	*pOpData*	Structure containing all the data needed to perform the operation. The client code allocates the memory for this structure. This component takes ownership of the memory until it is returned in the callback.
out	*pSignStatus*	In synchronous mode, the multiply output is valid (CPA_TRUE) or the output is invalid (CPA_FALSE).
out	*pR*	ECDSA message signature r.

Return values

CPA_STATUS_SUCCESS	Function executed successfully.
CPA_STATUS_FAIL	Function failed.
CPA_STATUS_RETRY	Resubmit the request.
CPA_STATUS_INVALID_PARAM	Invalid parameter passed in.
CPA_STATUS_RESOURCE	Error related to system resources.
CPA_STATUS_RESTARTING	API implementation is restarting. Resubmit the request.
CPA_STATUS_UNSUPPORTED	Function is not supported.

Precondition

The component has been initialized via cpaCyStartInstance function.

Postcondition

None

Note

When pCb is non-NULL an asynchronous callback is generated in response to this function call. For optimal performance, data pointers SHOULD be 8-byte aligned.

See also

None

4.14.3.2 cpaCyEcdsaSignS()

```
CpaStatus cpaCyEcdsaSignS (
              const CpaInstanceHandle instanceHandle,
              const CpaCyEcdsaGenSignCbFunc pCb,
              void * pCallbackTag,
              const CpaCyEcdsaSignSOpData * pOpData,
              CpaBoolean * pSignStatus,
              CpaFlatBuffer * pS )
```

Generate ECDSA Signature S.

Description:

This function generates ECDSA Signature S as per ANSI X9.62 2005 section 7.3.

Context:

When called as an asynchronous function it cannot sleep. It can be executed in a context that does not permit sleeping. When called as a synchronous function it may sleep. It MUST NOT be executed in a context that DOES NOT permit sleeping.

Assumptions:

None

Side-Effects:

None

Blocking:

Yes when configured to operate in synchronous mode.

Reentrant:

No

Thread-safe:

Yes

Parameters

in	*instanceHandle*	Instance handle.
in	*pCb*	Callback function pointer. If this is set to a NULL value the function will operate synchronously.
in	*pCallbackTag*	User-supplied value to help identify request.
in	*pOpData*	Structure containing all the data needed to perform the operation. The client code allocates the memory for this structure. This component takes ownership of the memory until it is returned in the callback.
out	*pSignStatus*	In synchronous mode, the multiply output is valid (CPA_TRUE) or the output is invalid (CPA_FALSE).
out	*pS*	ECDSA message signature s.

Return values

CPA_STATUS_SUCCESS	Function executed successfully.
CPA_STATUS_FAIL	Function failed.
CPA_STATUS_RETRY	Resubmit the request.
CPA_STATUS_INVALID_PARAM	Invalid parameter passed in.
CPA_STATUS_RESOURCE	Error related to system resources.
CPA_STATUS_RESTARTING	API implementation is restarting. Resubmit the request.
CPA_STATUS_UNSUPPORTED	Function is not supported.

Precondition

The component has been initialized via cpaCyStartInstance function.

Postcondition

None

Note

When pCb is non-NULL an asynchronous callback is generated in response to this function call. For optimal performance, data pointers SHOULD be 8-byte aligned.

See also

None

4.14.3.3 cpaCyEcdsaSignRS()

```
CpaStatus cpaCyEcdsaSignRS (
            const CpaInstanceHandle instanceHandle,
            const CpaCyEcdsaSignRSCbFunc pCb,
            void * pCallbackTag,
            const CpaCyEcdsaSignRSOpData * pOpData,
            CpaBoolean * pSignStatus,
            CpaFlatBuffer * pR,
            CpaFlatBuffer * pS )
```

Generate ECDSA Signature R & S.

Description:

This function generates ECDSA Signature R & S as per ANSI X9.62 2005 section 7.3.

Context:

When called as an asynchronous function it cannot sleep. It can be executed in a context that does not permit sleeping. When called as a synchronous function it may sleep. It MUST NOT be executed in a context that DOES NOT permit sleeping.

Assumptions:

None

Side-Effects:

None

Blocking:

Yes when configured to operate in synchronous mode.

Reentrant:

No

Thread-safe:

Yes

Parameters

in	*instanceHandle*	Instance handle.
in	*pCb*	Callback function pointer. If this is set to a NULL value the function will operate synchronously.
in	*pCallbackTag*	User-supplied value to help identify request.
in	*pOpData*	Structure containing all the data needed to perform the operation. The client code allocates the memory for this structure. This component takes ownership of the memory until it is returned in the callback.
out	*pSignStatus*	In synchronous mode, the multiply output is valid (CPA_TRUE) or the output is invalid (CPA_FALSE).
out	*pR*	ECDSA message signature r.
out	*pS*	ECDSA message signature s.

Return values

CPA_STATUS_SUCCESS	Function executed successfully.
CPA_STATUS_FAIL	Function failed.
CPA_STATUS_RETRY	Resubmit the request.
CPA_STATUS_INVALID_PARAM	Invalid parameter passed in.
CPA_STATUS_RESOURCE	Error related to system resources.
CPA_STATUS_RESTARTING	API implementation is restarting. Resubmit the request.
CPA_STATUS_UNSUPPORTED	Function is not supported.

Precondition

The component has been initialized via cpaCyStartInstance function.

Postcondition

None

Note

When pCb is non-NULL an asynchronous callback is generated in response to this function call. For optimal performance, data pointers SHOULD be 8-byte aligned.

See also

None

4.14.3.4 cpaCyEcdsaVerify()

```
CpaStatus cpaCyEcdsaVerify (
            const CpaInstanceHandle instanceHandle,
            const CpaCyEcdsaVerifyCbFunc pCb,
            void * pCallbackTag,
            const CpaCyEcdsaVerifyOpData * pOpData,
            CpaBoolean * pVerifyStatus )
```

Verify ECDSA Public Key.

Description:

This function performs ECDSA Verify as per ANSI X9.62 2005 section 7.4.

A response status of ok (verifyStatus == CPA_TRUE) means that the signature was verified

Context:

When called as an asynchronous function it cannot sleep. It can be executed in a context that does not permit sleeping. When called as a synchronous function it may sleep. It MUST NOT be executed in a context that DOES NOT permit sleeping.

Assumptions:

None

Side-Effects:

None

Blocking:

Yes when configured to operate in synchronous mode.

Reentrant:

No

Thread-safe:

Yes

Parameters

in	*instanceHandle*	Instance handle.
in	*pCb*	Callback function pointer. If this is set to a NULL value the function will operate synchronously.
in	*pCallbackTag*	User-supplied value to help identify request.
in	*pOpData*	Structure containing all the data needed to perform the operation. The client code allocates the memory for this structure. This component takes ownership of the memory until it is returned in the callback.
out	*pVerifyStatus*	In synchronous mode, set to CPA_FALSE if the point is NOT on the curve or at infinity. Set to CPA_TRUE if the point is on the curve.

Return values

CPA_STATUS_SUCCESS	Function executed successfully.
CPA_STATUS_FAIL	Function failed.
CPA_STATUS_RETRY	Resubmit the request.
CPA_STATUS_INVALID_PARAM	Invalid parameter passed in.

Return values

CPA_STATUS_RESOURCE	Error related to system resources.
CPA_STATUS_RESTARTING	API implementation is restarting. Resubmit the request.
CPA_STATUS_UNSUPPORTED	Function is not supported.

Precondition

The component has been initialized via cpaCyStartInstance function.

Postcondition

None

Note

When pCb is non-NULL an asynchronous callback of type CpaCyEcdsaVerifyCbFunc is generated in response to this function call. For optimal performance, data pointers SHOULD be 8-byte aligned.

See also

CpaCyEcdsaVerifyOpData, CpaCyEcdsaVerifyCbFunc

4.14.3.5 cpaCyEcdsaQueryStats64()

```
CpaStatus cpaCyEcdsaQueryStats64 (
            const CpaInstanceHandle instanceHandle,
            CpaCyEcdsaStats64 * pEcdsaStats )
```

Query statistics for a specific ECDSA instance.

Description:

This function will query a specific instance of the ECDSA implementation for statistics. The user MUST allocate the CpaCyEcdsaStats64 structure and pass the reference to that structure into this function call. This function writes the statistic results into the passed in CpaCyEcdsaStats64 structure.

Note: statistics returned by this function do not interrupt current data processing and as such can be slightly out of sync with operations that are in progress during the statistics retrieval process.

Context:

This is a synchronous function and it can sleep. It MUST NOT be executed in a context that DOES NOT permit sleeping.

Assumptions:

None

Side-Effects:

None

Blocking:

This function is synchronous and blocking.

Reentrant:

No

Thread-safe:

Yes

Parameters

in	*instanceHandle*	Instance handle.
out	*pEcdsaStats*	Pointer to memory into which the statistics will be written.

Return values

CPA_STATUS_SUCCESS	Function executed successfully.
CPA_STATUS_FAIL	Function failed.
CPA_STATUS_INVALID_PARAM	Invalid parameter passed in.
CPA_STATUS_RESOURCE	Error related to system resources.
CPA_STATUS_RESTARTING	API implementation is restarting. Resubmit the request.
CPA_STATUS_UNSUPPORTED	Function is not supported.

Precondition

Component has been initialized.

Postcondition

None

Note

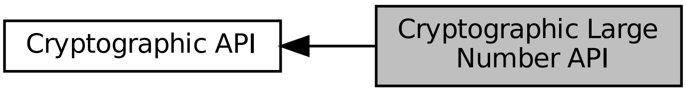
This function operates in a synchronous manner and no asynchronous callback will be generated.

See also

CpaCyEcdsaStats64

4.15 Cryptographic Large Number API

Collaboration diagram for Cryptographic Large Number API:



Data Structures

- struct _CpaCyLnModExpOpData
- struct _CpaCyLnModInvOpData
- struct _CpaCyLnStats
- struct _CpaCyLnStats64

Typedefs

- typedef struct _CpaCyLnModExpOpData CpaCyLnModExpOpData
- typedef struct _CpaCyLnModInvOpData CpaCyLnModInvOpData
- typedef struct _CpaCyLnStats CPA_DEPRECATED
- typedef struct _CpaCyLnStats64 CpaCyLnStats64

Functions

- CpaStatus cpaCyLnModExp (const CpaInstanceHandle instanceHandle, const CpaCyGenFlatBufCbFunc pLnModExpCb, void ∗pCallbackTag, const CpaCyLnModExpOpData ∗pLnModExpOpData, CpaFlatBuffer ∗pResult)
- CpaStatus cpaCyLnModInv (const CpaInstanceHandle instanceHandle, const CpaCyGenFlatBufCbFunc pLnModInvCb, void ∗pCallbackTag, const CpaCyLnModInvOpData ∗pLnModInvOpData, CpaFlatBuffer ∗pResult)
- CpaStatus CPA_DEPRECATED cpaCyLnStatsQuery (const CpaInstanceHandle instanceHandle, struct _CpaCyLnStats ∗pLnStats)
- CpaStatus cpaCyLnStatsQuery64 (const CpaInstanceHandle instanceHandle, CpaCyLnStats64 ∗pLnStats)

4.15.1 Detailed Description

File: cpa_cy_ln.h

Description:

These functions specify the Cryptographic API for Large Number Operations.

Note

Large numbers are represented on the QuickAssist API using octet strings, stored in structures of type CpaFlatBuffer. These octet strings are encoded as described by PKCS#1 v2.1, section 4, which is consistent with ASN.1 syntax. The following text summarizes this. Any exceptions to this encoding are specified on the specific data structure or function to which the exception applies.

An n-bit number, N, has a value in the range 2^(n-1) through 2^(n)-1. In other words, its most significant bit, bit n-1 (where bit-counting starts from zero) MUST be set to 1. We can also state that the bit-length n of a number N is defined by n = floor(log2(N))+1.

The buffer, b, in which an n-bit number N is stored, must be "large enough". In other words, b.dataLenInBytes must be at least minLenInBytes = ceiling(n/8).

The number is stored in a "big endian" format. This means that the least significant byte (LSB) is b[b.dataLenInBytes-1], while the most significant byte (MSB) is b[b.dataLenInBytes-minLenInBytes]. In the case where the buffer is "exactly" the right size, then the MSB is b[0]. Otherwise, all bytes from b[0] up to the MSB MUST be set to 0x00.

The largest bit-length we support today is 8192 bits. In other words, we can deal with numbers up to a value of (2^8192)-1.

4.15.2 Typedef Documentation

4.15.2.1 CpaCyLnModExpOpData

```
typedef struct _CpaCyLnModExpOpData CpaCyLnModExpOpData
```

Modular Exponentiation Function Operation Data.

Description:

This structure lists the different items that are required in the cpaCyLnModExp function. The client MUST allocate the memory for this structure. When the structure is passed into the function, ownership of the memory passes to the function. Ownership of the memory returns to the client when this structure is returned in the callback. The operation size in bits is equal to the size of whichever of the following is largest: the modulus, the base or the exponent.

Note

If the client modifies or frees the memory referenced in this structure after it has been submitted to the cpaCyLnModExp function, and before it has been returned in the callback, undefined behavior will result.

The values of the base, the exponent and the modulus MUST all be less than 2^8192, and the modulus must not be equal to zero.

4.15.2.2 CpaCyLnModInvOpData

```
typedef struct _CpaCyLnModInvOpData CpaCyLnModInvOpData
```

Modular Inversion Function Operation Data.

Description:

This structure lists the different items that are required in the function cpaCyLnModInv. The client MUST allocate the memory for this structure. When the structure is passed into the function, ownership of the memory passes to the function. Ownership of the memory returns to the client when this structure is returned in the callback.

Note

If the client modifies or frees the memory referenced in this structure after it has been submitted to the cpaCyLnModInv function, and before it has been returned in the callback, undefined behavior will result.

Note that the values of A and B MUST NOT both be even numbers, and both MUST be less than 2^8192.

4.15.2.3 CPA_DEPRECATED

```
typedef struct _CpaCyLnStats CPA_DEPRECATED
```

Look Aside Cryptographic large number Statistics.

Deprecated As of v1.3 of the Crypto API, this structure has been deprecated, replaced by CpaCyLnStats64.

Description:

This structure contains statistics on the Look Aside Cryptographic large number operations. Statistics are set to zero when the component is initialized, and are collected per instance.

4.15.2.4 CpaCyLnStats64

```
typedef struct _CpaCyLnStats64 CpaCyLnStats64
```

Look Aside Cryptographic large number Statistics.

Description:

This structure contains statistics on the Look Aside Cryptographic large number operations. Statistics are set to zero when the component is initialized, and are collected per instance.

4.15.3 Function Documentation

4.15.3.1 cpaCyLnModExp()

```
CpaStatus cpaCyLnModExp (
            const CpaInstanceHandle instanceHandle,
            const CpaCyGenFlatBufCbFunc pLnModExpCb,
            void * pCallbackTag,
            const CpaCyLnModExpOpData * pLnModExpOpData,
            CpaFlatBuffer * pResult )
```

Perform modular exponentiation operation.

Description:

This function performs modular exponentiation. It computes the following result based on the inputs:

result = (base ^ exponent) mod modulus

Context:

When called as an asynchronous function it cannot sleep. It can be executed in a context that does not permit sleeping. When called as a synchronous function it may sleep. It MUST NOT be executed in a context that DOES NOT permit sleeping.

Assumptions:

None

Side-Effects:

None

Reentrant:

No

Thread-safe:

Yes

Parameters

in	*instanceHandle*	Instance handle.
in	*pLnModExpCb*	Pointer to callback function to be invoked when the operation is complete.
in	*pCallbackTag*	Opaque User Data for this specific call. Will be returned unchanged in the callback.
in	*pLnModExpOpData*	Structure containing all the data needed to perform the LN modular exponentiation operation. The client code allocates the memory for this structure. This component takes ownership of the memory until it is returned in the callback.
out	*pResult*	Pointer to a flat buffer containing a pointer to memory allocated by the client into which the result will be written. The size of the memory required MUST be larger than or equal to the size required to store the modulus. On invocation the callback function will contain this parameter in the pOut parameter.

Return values

CPA_STATUS_SUCCESS	Function executed successfully.
CPA_STATUS_FAIL	Function failed.
CPA_STATUS_RETRY	Resubmit the request.
CPA_STATUS_INVALID_PARAM	Invalid parameter passed in.
CPA_STATUS_RESOURCE	Error related to system resources.
CPA_STATUS_RESTARTING	API implementation is restarting. Resubmit the request.
CPA_STATUS_UNSUPPORTED	Function is not supported.

Precondition

The component has been initialized.

Postcondition

None

Note

When pLnModExpCb is non null, an asynchronous callback of type CpaCyLnModExpCbFunc is generated in response to this function call. Any errors generated during processing are reported in the structure returned in the callback.

See also

CpaCyLnModExpOpData, CpaCyGenFlatBufCbFunc

4.15.3.2 cpaCyLnModInv()

```
CpaStatus cpaCyLnModInv (
            const CpaInstanceHandle instanceHandle,
            const CpaCyGenFlatBufCbFunc pLnModInvCb,
            void * pCallbackTag,
            const CpaCyLnModInvOpData * pLnModInvOpData,
            CpaFlatBuffer * pResult )
```

Perform modular inversion operation.

Description:

This function performs modular inversion. It computes the following result based on the inputs:

result = (1/A) mod B.

Context:

When called as an asynchronous function it cannot sleep. It can be executed in a context that does not permit sleeping. When called as a synchronous function it may sleep. It MUST NOT be executed in a context that DOES NOT permit sleeping.

Assumptions:

None

Side-Effects:

None

Reentrant:

No

Thread-safe:

Yes

Parameters

in	*instanceHandle*	Instance handle.
in	*pLnModInvCb*	Pointer to callback function to be invoked when the operation is complete.
in	*pCallbackTag*	Opaque User Data for this specific call. Will be returned unchanged in the callback.
in	*pLnModInvOpData*	Structure containing all the data needed to perform the LN modular inversion operation. The client code allocates the memory for this structure. This component takes ownership of the memory until it is returned in the callback.
out	*pResult*	Pointer to a flat buffer containing a pointer to memory allocated by the client into which the result will be written. The size of the memory required MUST be larger than or equal to the size required to store the modulus. On invocation the callback function will contain this parameter in the pOut parameter.

Return values

CPA_STATUS_SUCCESS	Function executed successfully.
CPA_STATUS_FAIL	Function failed.
CPA_STATUS_RETRY	Resubmit the request.
CPA_STATUS_INVALID_PARAM	Invalid parameter passed in.
CPA_STATUS_RESOURCE	Error related to system resources.
CPA_STATUS_RESTARTING	API implementation is restarting. Resubmit the request.
CPA_STATUS_UNSUPPORTED	Function is not supported.

Precondition

The component has been initialized.

Postcondition

None

Note

When pLnModInvCb is non null, an asynchronous callback of type CpaCyLnModInvCbFunc is generated in response to this function call. Any errors generated during processing are reported in the structure returned in the callback.

See also

CpaCyLnModInvOpData, CpaCyGenFlatBufCbFunc

4.15.3.3 cpaCyLnStatsQuery()

```
CpaStatus CPA_DEPRECATED cpaCyLnStatsQuery (
            const CpaInstanceHandle instanceHandle,
            struct _CpaCyLnStats * pLnStats )
```

Query statistics for large number operations

Deprecated As of v1.3 of the Crypto API, this function has been deprecated, replaced by cpaCyLnStatsQuery64().

Description:

This function will query a specific instance handle for large number statistics. The user MUST allocate the CpaCyLnStats structure and pass the reference to that structure into this function call. This function writes the statistic results into the passed in CpaCyLnStats structure.

Note: statistics returned by this function do not interrupt current data processing and as such can be slightly out of sync with operations that are in progress during the statistics retrieval process.

Context:

This is a synchronous function and it can sleep. It MUST NOT be executed in a context that DOES NOT permit sleeping.

Assumptions:

None

Side-Effects:

None

Reentrant:

No

Thread-safe:

Yes

Parameters

in	*instanceHandle*	Instance handle.
out	*pLnStats*	Pointer to memory into which the statistics will be written.

Return values

CPA_STATUS_SUCCESS	Function executed successfully.
CPA_STATUS_FAIL	Function failed.
CPA_STATUS_INVALID_PARAM	Invalid parameter passed in.
CPA_STATUS_RESOURCE	Error related to system resources.
CPA_STATUS_RESTARTING	API implementation is restarting. Resubmit the request.
CPA_STATUS_UNSUPPORTED	Function is not supported.

Precondition

Acceleration Services unit has been initialized.

Postcondition

None

Note

This function operates in a synchronous manner and no asynchronous callback will be generated.

See also

CpaCyLnStats

4.15.3.4 cpaCyLnStatsQuery64()

```
CpaStatus cpaCyLnStatsQuery64 (
            const CpaInstanceHandle instanceHandle,
            CpaCyLnStats64 * pLnStats )
```

Query statistics (64-bit version) for large number operations

Description:

This function will query a specific instance handle for the 64-bit version of the large number statistics. The user MUST allocate the CpaCyLnStats64 structure and pass the reference to that structure into this function call. This function writes the statistic results into the passed in CpaCyLnStats64 structure.

Note: statistics returned by this function do not interrupt current data processing and as such can be slightly out of sync with operations that are in progress during the statistics retrieval process.

Context:

This is a synchronous function and it can sleep. It MUST NOT be executed in a context that DOES NOT permit sleeping.

Assumptions:

None

Side-Effects:

None

Reentrant:

No

Thread-safe:

Yes

Parameters

`in`	*instanceHandle*	Instance handle.
`out`	*pLnStats*	Pointer to memory into which the statistics will be written.

Return values

CPA_STATUS_SUCCESS	Function executed successfully.
CPA_STATUS_FAIL	Function failed.
CPA_STATUS_INVALID_PARAM	Invalid parameter passed in.
CPA_STATUS_RESOURCE	Error related to system resources.
CPA_STATUS_RESTARTING	API implementation is restarting. Resubmit the request.
CPA_STATUS_UNSUPPORTED	Function is not supported.

Precondition

Acceleration Services unit has been initialized.

Postcondition

None

Note

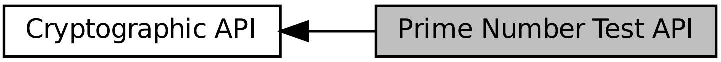
This function operates in a synchronous manner and no asynchronous callback will be generated.

See also

CpaCyLnStats

4.16 Prime Number Test API

Collaboration diagram for Prime Number Test API:



Data Structures

- struct _CpaCyPrimeTestOpData
- struct _CpaCyPrimeStats
- struct _CpaCyPrimeStats64

Typedefs

- typedef struct _CpaCyPrimeTestOpData CpaCyPrimeTestOpData
- typedef struct _CpaCyPrimeStats CPA_DEPRECATED
- typedef struct _CpaCyPrimeStats64 CpaCyPrimeStats64
- typedef void(∗ CpaCyPrimeTestCbFunc) (void ∗pCallbackTag, CpaStatus status, void ∗pOpData, CpaBoolean testPassed)

Functions

- CpaStatus cpaCyPrimeTest (const CpaInstanceHandle instanceHandle, const CpaCyPrimeTestCbFunc pCb, void ∗pCallbackTag, const CpaCyPrimeTestOpData ∗pOpData, CpaBoolean ∗pTestPassed)

4.16.1 Detailed Description

File: cpa_cy_prime.h

Description:

These functions specify the API for the prime number test operations.

For prime number generation, this API SHOULD be used in conjunction with the Deterministic Random Bit Generation API (cpaCyDrbg).

Note

Large numbers are represented on the QuickAssist API as described in the Large Number API (Cryptographic Large Number API).

In addition, the bit length of large numbers passed to the API MUST NOT exceed 576 bits for Elliptic Curve operations.

4.16.2 Typedef Documentation

4.16.2.1 CpaCyPrimeTestOpData

```
typedef struct _CpaCyPrimeTestOpData CpaCyPrimeTestOpData
```

Prime Test Operation Data.

Description:

This structure contains the operation data for the cpaCyPrimeTest function. The client MUST allocate the memory for this structure and the items pointed to by this structure. When the structure is passed into the function, ownership of the memory passes to the function. Ownership of the memory returns to the client when this structure is returned in the callback function.

All values in this structure are required to be in Most Significant Byte first order, e.g. primeCandidate.pData[0] = MSB.

All numbers MUST be stored in big-endian order.

Note

If the client modifies or frees the memory referenced in this structure after it has been submitted to the cpaCyPrimeTest function, and before it has been returned in the callback, undefined behavior will result.

See also

cpaCyPrimeTest()

4.16.2.2 CPA_DEPRECATED

```
typedef struct _CpaCyPrimeStats CPA_DEPRECATED
```

Prime Number Test Statistics.

Deprecated As of v1.3 of the Crypto API, this structure has been deprecated, replaced by CpaCyPrimeStats64.

Description:

This structure contains statistics on the prime number test operations. Statistics are set to zero when the component is initialized, and are collected per instance.

4.16.2.3 CpaCyPrimeStats64

```
typedef struct _CpaCyPrimeStats64 CpaCyPrimeStats64
```

Prime Number Test Statistics (64-bit version).

Description:

This structure contains a 64-bit version of the statistics on the prime number test operations. Statistics are set to zero when the component is initialized, and are collected per instance.

4.16.2.4 CpaCyPrimeTestCbFunc

```
typedef void(* CpaCyPrimeTestCbFunc) (void *pCallbackTag, CpaStatus status, void *pOpData,
CpaBoolean testPassed)
```

Definition of callback function invoked for cpaCyPrimeTest requests.

Description:

This is the prototype for the cpaCyPrimeTest callback function.

Context:

This callback function can be executed in a context that DOES NOT permit sleeping to occur.

Assumptions:

None

Side-Effects:

None

Reentrant:

No

Thread-safe:

Yes

Parameters

`in`	*pCallbackTag*	User-supplied value to help identify request.
`in`	*status*	Status of the operation. Valid values are CPA_STATUS_SUCCESS, CPA_STATUS_FAIL and CPA_STATUS_UNSUPPORTED.
`in`	*pOpData*	Opaque pointer to the Operation data pointer supplied in request.
`in`	*testPassed*	A value of CPA_TRUE means the prime candidate is probably prime.

Return values

None	

Precondition

Component has been initialized.

Postcondition

None

Note

None

See also

cpaCyPrimeTest()

Definition at line 201 of file cpa_cy_prime.h.

4.16.3 Function Documentation

4.16.3.1 cpaCyPrimeTest()

```
CpaStatus cpaCyPrimeTest (
            const CpaInstanceHandle instanceHandle,
            const CpaCyPrimeTestCbFunc pCb,
            void * pCallbackTag,
            const CpaCyPrimeTestOpData * pOpData,
            CpaBoolean * pTestPassed )
```

Prime Number Test Function.

Description:

This function will test probabilistically if a number is prime. Refer to ANSI X9.80 2005 for details. The primality result will be returned in the asynchronous callback.

The following combination of GCD, Fermat, Miller-Rabin, and Lucas testing is supported: (up to 1x GCD) + (up to 1x Fermat) + (up to 50x Miller-Rabin rounds) + (up to 1x Lucas) For example: (1x GCD) + (25x Miller-Rabin) + (1x Lucas); (1x GCD) + (1x Fermat); (50x Miller-rabin);

Tests are always performed in order of increasing complexity, for example GCD first, then Fermat, then Miller-Rabin, and finally Lucas.

For all of the primality tests, the following prime number "sizes" (length in bits) are supported: all sizes up to and including 512 bits, as well as sizes 768, 1024, 1536, 2048, 3072 and 4096.

Candidate prime numbers MUST match these sizes accordingly, with leading zeroes present where necessary.

When this prime number test is used in conjunction with combined Miller-Rabin and Lucas tests, it may be used as a means of performing a self test operation on the random data generator.

A response status of ok (pass == CPA_TRUE) means all requested primality tests passed, and the prime candidate is probably prime (the exact probability depends on the primality tests requested). A response status of not ok (pass == CPA_FALSE) means one of the requested primality tests failed (the prime candidate has been found to be composite).

Context:

When called as an asynchronous function it cannot sleep. It can be executed in a context that does not permit sleeping. When called as a synchronous function it may sleep. It MUST NOT be executed in a context that DOES NOT permit sleeping.

Assumptions:

None

Side-Effects:

None

Blocking:

Yes when configured to operate in synchronous mode.

Reentrant:

No

Thread-safe:

Yes

Parameters

in	*instanceHandle*	Instance handle.
in	*pCb*	Callback function pointer. If this is set to a NULL value the function will operate synchronously.
in	*pCallbackTag*	User-supplied value to help identify request.
in	*pOpData*	Structure containing all the data needed to perform the operation. The client code allocates the memory for this structure. This component takes ownership of the memory until it is returned in the callback.
out	*pTestPassed*	A value of CPA_TRUE means the prime candidate is probably prime.

Return values

CPA_STATUS_SUCCESS	Function executed successfully.
CPA_STATUS_FAIL	Function failed.
CPA_STATUS_RETRY	Resubmit the request.
CPA_STATUS_INVALID_PARAM	Invalid parameter passed in.
CPA_STATUS_RESOURCE	Error related to system resources.
CPA_STATUS_RESTARTING	API implementation is restarting. Resubmit the request.
CPA_STATUS_UNSUPPORTED	Function is not supported.

Precondition

The component has been initialized via cpaCyStartInstance function.

Postcondition

None

Note

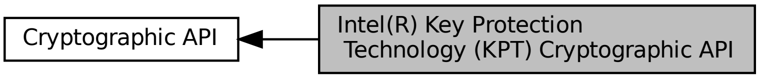
When pCb is non-NULL an asynchronous callback of type CpaCyPrimeTestCbFunc is generated in response to this function call. For optimal performance, data pointers SHOULD be 8-byte aligned.

See also

CpaCyPrimeTestOpData, CpaCyPrimeTestCbFunc

4.17 Intel(R) Key Protection Technology (KPT) Cryptographic API

Collaboration diagram for Intel(R) Key Protection Technology (KPT) Cryptographic API:



Data Structures

- struct CpaCyKptValidationKey_t
- struct CpaCyKptLoadKey_t
- struct CpaCyKptUnwrapContext_t
- struct CpaCyKptRsaPrivateKeyRep1_t
- struct CpaCyKptRsaPrivateKeyRep2_t
- struct CpaCyKptRsaPrivateKey_t
- struct CpaCyKptRsaDecryptOpData_t
- struct CpaCyKptEcdsaSignRSOpData_t

Macros

- #define CPA_CY_RSA3K_SIG_SIZE_INBYTES
- #define CPA_CY_KPT_MAX_IV_LENGTH
- #define CPA_CY_KPT_MAX_AAD_LENGTH

Typedefs

- typedef Cpa64U CpaCyKptHandle
- typedef enum CpaCyKptKeyManagementStatus_t CpaCyKptKeyManagementStatus
- typedef struct CpaCyKptValidationKey_t CpaCyKptValidationKey
- typedef enum CpaCyKptWrappingKeyType_t CpaCyKptWrappingKeyType
- typedef struct CpaCyKptLoadKey_t CpaCyKptLoadKey
- typedef struct CpaCyKptUnwrapContext_t CpaCyKptUnwrapContext
- typedef struct CpaCyKptRsaPrivateKeyRep1_t CpaCyKptRsaPrivateKeyRep1
- typedef struct CpaCyKptRsaPrivateKeyRep2_t CpaCyKptRsaPrivateKeyRep2
- typedef struct CpaCyKptRsaPrivateKey_t CpaCyKptRsaPrivateKey
- typedef struct CpaCyKptRsaDecryptOpData_t CpaCyKptRsaDecryptOpData
- typedef struct CpaCyKptEcdsaSignRSOpData_t CpaCyKptEcdsaSignRSOpData

Enumerations

- enum CpaCyKptKeyManagementStatus_t
- enum CpaCyKptWrappingKeyType_t

Functions

- CpaStatus cpaCyKptQueryIssuingKeys (const CpaInstanceHandle instanceHandle, CpaFlatBuffer ∗pPublicX509IssueCert, CpaCyKptKeyManagementStatus ∗pKptStatus)
- CpaStatus cpaCyKptQueryDeviceCredentials (const CpaInstanceHandle instanceHandle, CpaCyKptValidationKey ∗pDevCredential, CpaCyKptKeyManagementStatus ∗pKptStatus)
- CpaStatus cpaCyKptLoadKey (CpaInstanceHandle instanceHandle, CpaCyKptLoadKey ∗pSWK, CpaCyKptHandle ∗keyHandle, CpaCyKptKeyManagementStatus ∗pKptStatus)
- CpaStatus cpaCyKptDeleteKey (CpaInstanceHandle instanceHandle, CpaCyKptHandle keyHandle, CpaCyKptKeyManagementStatus ∗pKptStatus)
- CpaStatus cpaCyKptRsaDecrypt (const CpaInstanceHandle instanceHandle, const CpaCyGenFlatBufCbFunc pRsaDecryptCb, void ∗pCallbackTag, const CpaCyKptRsaDecryptOpData ∗pDecryptOpData, CpaFlatBuffer ∗pOutputData, CpaCyKptUnwrapContext ∗pKptUnwrapContext)
- CpaStatus cpaCyKptEcdsaSignRS (const CpaInstanceHandle instanceHandle, const CpaCyEcdsaSignRSCbFunc pCb, void ∗pCallbackTag, const CpaCyKptEcdsaSignRSOpData ∗pOpData, CpaBoolean ∗pSignStatus, CpaFlatBuffer ∗pR, CpaFlatBuffer ∗pS, CpaCyKptUnwrapContext ∗pKptUnwrapContext)

4.17.1 Detailed Description

File: cpa_cy_kpt.h

Description:

These functions specify the APIs for Key Protection Technology (KPT) Cryptographic services.

Note

These functions implement the KPT Cryptographic API. This API is experimental and subject to change.

4.17.2 Macro Definition Documentation

4.17.2.1 CPA_CY_RSA3K_SIG_SIZE_INBYTES

```
#define CPA_CY_RSA3K_SIG_SIZE_INBYTES
```

PKCS#1 v2.2 RSA-3K signature output length in bytes.

See also

CpaCyKptValidationKey

Definition at line 89 of file cpa_cy_kpt.h.

4.17.2.2 CPA_CY_KPT_MAX_IV_LENGTH

```
#define CPA_CY_KPT_MAX_IV_LENGTH
```

Max length of initialization vector

Description:

Defines the permitted max iv length in bytes that may be used in private key wrapping/unwrapping.For AEC-GCM, iv length is 12 bytes.

See also

cpaCyKptUnwrapContext

Definition at line 153 of file cpa_cy_kpt.h.

4.17.2.3 CPA_CY_KPT_MAX_AAD_LENGTH

```
#define CPA_CY_KPT_MAX_AAD_LENGTH
```

Max length of Additional Authenticated Data

Description:

Defines the permitted max aad length in bytes that may be used in private key wrapping/unwrapping.

See also

cpaCyKptUnwrapContext

Definition at line 166 of file cpa_cy_kpt.h.

4.17.3 Typedef Documentation

4.17.3.1 CpaCyKptHandle

```
typedef Cpa64U CpaCyKptHandle
```

KPT wrapping key handle

Description:

Handle to a unique wrapping key in wrapping key table. Application creates it in KPT key transfer phase and maintains it for KPT Crypto service. For each KPT Crypto service API invocation, this handle will be used to get a SWK(Symmetric Wrapping Key) to unwrap WPK(Wrapped Private Key) before performing the requested crypto service.

Definition at line 55 of file cpa_cy_kpt.h.

4.17.3.2 CpaCyKptKeyManagementStatus

```
typedef enum CpaCyKptKeyManagementStatus_t CpaCyKptKeyManagementStatus
```

Return Status

Description:

This enumeration lists all the possible return status after completing KPT APIs.

4.17.3.3 CpaCyKptValidationKey

```
typedef struct CpaCyKptValidationKey_t CpaCyKptValidationKey
```

KPT device credentials key certificate

Description:

This structure defines the key format for use with KPT.

See also

cpaCyKptQueryDeviceCredentials

4.17.3.4 CpaCyKptWrappingKeyType

```
typedef enum CpaCyKptWrappingKeyType_t CpaCyKptWrappingKeyType
```

Cipher algorithms used to generate a wrapped private key (WPK) from the clear private key.

Description:

This enumeration lists supported cipher algorithms and modes.

4.17.3.5 CpaCyKptLoadKey

```
typedef struct CpaCyKptLoadKey_t CpaCyKptLoadKey
```

KPT Loading key format specification.

Description:

This structure defines the format of the symmetric wrapping key to be loaded into KPT. Application sets these parameters through the cpaCyKptLoadKey calls.

4.17.3.6 CpaCyKptUnwrapContext

```
typedef struct CpaCyKptUnwrapContext_t CpaCyKptUnwrapContext
```

File: cpa_cy_kpt.h

Structure of KPT unwrapping context.

Description:

This structure is a parameter of KPT crypto APIs, it contains data relating to KPT WPK unwrapping, the application needs to fill in this information.

4.17.3.7 CpaCyKptRsaPrivateKeyRep1

```
typedef struct CpaCyKptRsaPrivateKeyRep1_t CpaCyKptRsaPrivateKeyRep1
```

File: cpa_cy_kpt.h

RSA Private Key Structure For Representation 1.

Description:

This structure contains the first representation that can be used for describing the RSA private key, represented by the tuple of the modulus (N) and the private exponent (D). The representation is encrypted as follows: Encrypt - AES-256-GCM (Key, AAD, Input) "||" - denotes concatenation Key = SWK AAD = DER(OID) Input = (D || N) Encrypt (SWK, AAD, (D || N)) Output (AuthTag, (D || N)') EncryptedRSAKey = (D || N)'

privateKey = (EncryptedRSAKey || AuthTag)

OID's that shall be supported by KPT implementation: OID DER(OID) 1.2.840.113549.1.1 06 08 2A 86 48 86 F7 0D 01 01

Permitted lengths for N and D are:

- 512 bits (64 bytes),
- 1024 bits (128 bytes),
- 1536 bits (192 bytes),
- 2048 bits (256 bytes),
- 3072 bits (384 bytes),
- 4096 bits (512 bytes), or
- 8192 bits (1024 bytes).

AuthTag is 128 bits (16 bytes)

Note

It is important that the value D is big enough. It is STRONGLY recommended that this value is at least half the length of the modulus N to protect against the Wiener attack.

4.17.3.8 CpaCyKptRsaPrivateKeyRep2

```
typedef struct CpaCyKptRsaPrivateKeyRep2_t CpaCyKptRsaPrivateKeyRep2
```

File: cpa_cy_kpt.h

KPT RSA Private Key Structure For Representation 2.

Description:

This structure contains the second representation that can be used for describing the RSA private key. The quintuple of p, q, dP, dQ, and qInv (explained below and in the spec) are required for the second representation. For KPT the parameters are Encrypted with the assoicated SWK as follows: Encrypt - AES-256-GCM (Key, AAD, Input) "||" - denotes concatenation Key = SWK AAD = DER(OID) Input = (P || Q || dP || dQ || Qinv || publicExponentE) Expanded Description: Encrypt (SWK, AAD, (P || Q || dP || dQ || Qinv || publicExponentE)) EncryptedRSAKey = (P || Q || dP || dQ || Qinv || publicExponentE)' Output (AuthTag, EncryptedRSAKey)

privateKey = EncryptedRSAKey || AuthTag

OID's that shall be supported by KPT implementation: OID DER(OID) 1.2.840.113549.1.1 06 08 2A 86 48 86 F7 0D 01 01

All of the encrypted parameters will be of equal size. The length of each will be equal to keySize in bytes/2. For example for a key size of 256 Bytes (2048 bits), the length of P, Q, dP, dQ, and Qinv are all 128 Bytes, plus the publicExponentE of 256 Bytes, giving a total size for EncryptedRSAKey of 896 Bytes.

AuthTag is 128 bits (16 bytes)

Permitted Key Sizes are:

- 512 bits (64 bytes),
- 1024 bits (128 bytes),
- 1536 bits (192 bytes),
- 2048 bits (256 bytes),
- 3072 bits (384 bytes),
- 4096 bits (512 bytes), or
- 8192 bits (1024 bytes).

4.17.3.9 CpaCyKptRsaPrivateKey

```
typedef struct CpaCyKptRsaPrivateKey_t CpaCyKptRsaPrivateKey
```

File: cpa_cy_kpt.h

RSA Private Key Structure.

Description:

This structure contains the two representations that can be used for describing the RSA private key. The privateKeyRepType will be used to identify which representation is to be used. Typically, using the second representation results in faster decryption operations.

4.17.3.10 CpaCyKptRsaDecryptOpData

```
typedef struct CpaCyKptRsaDecryptOpData_t CpaCyKptRsaDecryptOpData
```

File: cpa_cy_kpt.h

KPT RSA Decryption Primitive Operation Data

Description:

This structure lists the different items that are required in the cpaCyKptRsaDecrypt function. As the RSA decryption primitive and signature primitive operations are mathematically identical this structure may also be used to perform an RSA signature primitive operation. When performing an RSA decryption primitive operation, the input data is the cipher text and the output data is the message text. When performing an RSA signature primitive operation, the input data is the message and the output data is the signature. The client MUST allocate the memory for this structure. When the structure is passed into the function, ownership of the memory passes to he function. Ownership of the memory returns to the client when this structure is returned in the CpaCyGenFlatBufCbFunc callback function.

Note

If the client modifies or frees the memory referenced in this structure after it has been submitted to the cpaCyKptRsaDecrypt function, and before it has been returned in the callback, undefined behavior will result. All values in this structure are required to be in Most Significant Byte first order, e.g. inputData.pData[0] = MSB.

4.17.3.11 CpaCyKptEcdsaSignRSOpData

```
typedef struct CpaCyKptEcdsaSignRSOpData_t CpaCyKptEcdsaSignRSOpData
```

File: cpa_cy_kpt.h

KPT ECDSA Sign R & S Operation Data.

Description:

This structure contains the operation data for the cpaCyKptEcdsaSignRS function. The client MUST allocate the memory for this structure and the items pointed to by this structure. When the structure is passed into the function, ownership of the memory passes to the function. Ownership of the memory returns to the client when this structure is returned in the callback function. This key structure is encrypted when passed into cpaCyKptEcdsaSignRS Encrypt - AES-256-GCM (Key, AAD, Input) "||" - denotes concatenation

Key = SWK AAD = DER(OID) Input = (d) Encrypt (SWK, AAD, (d)) Output (AuthTag, EncryptedECKey)

privatekey == EncryptedECKey || AuthTag

OID's that shall be supported by KPT implementation: Curve OID DER(OID) secp256r1 1.2.840.10045.3.1.7 06 08 2A 86 48 CE 3D 03 01 07 secp384r1 1.3.132.0.34 06 05 2B 81 04 00 22 secp521r1 1.3.132.0.35 06 05 2B 81 04 00 23

Expected private key (d) sizes: secp256r1 256 bits secp384r1 384 bits secp521r1 576 bits (rounded up to a multiple of 64-bit quadword)

AuthTag is 128 bits (16 bytes)

For optimal performance all data buffers SHOULD be 8-byte aligned.

Note

If the client modifies or frees the memory referenced in this structure after it has been submitted to the cpaCyKptEcdsaSignRS function, and before it has been returned in the callback, undefined behavior will result.

See also

cpaCyEcdsaSignRS()

4.17.4 Enumeration Type Documentation

4.17.4.1 CpaCyKptKeyManagementStatus_t

```
enum CpaCyKptKeyManagementStatus_t
```

Return Status

Description:

This enumeration lists all the possible return status after completing KPT APIs.

Enumerator

CPA_CY_KPT_SUCCESS	Generic success status for all KPT wrapping key handling functions
CPA_CY_KPT_LOADKEY_FAIL_QUOTA_EXCEEDED_PER_VFID	SWK count exceeds the configured maxmium value per VFID
CPA_CY_KPT_LOADKEY_FAIL_QUOTA_EXCEEDED_PER_PASID	SWK count exceeds the configured maxmium value per PASID
CPA_CY_KPT_LOADKEY_FAIL_QUOTA_EXCEEDED	SWK count exceeds the configured maxmium value when not scoped to VFID or PASID
CPA_CY_KPT_SWK_FAIL_NOT_FOUND	Unable to find SWK entry by handle

Definition at line 66 of file cpa_cy_kpt.h.

4.17.4.2 CpaCyKptWrappingKeyType_t

```
enum CpaCyKptWrappingKeyType_t
```

Cipher algorithms used to generate a wrapped private key (WPK) from the clear private key.

Description:

This enumeration lists supported cipher algorithms and modes.

Definition at line 119 of file cpa_cy_kpt.h.

4.17.5 Function Documentation

4.17.5.1 cpaCyKptQueryIssuingKeys()

```
CpaStatus cpaCyKptQueryIssuingKeys (
            const CpaInstanceHandle instanceHandle,
            CpaFlatBuffer * pPublicX509IssueCert,
            CpaCyKptKeyManagementStatus * pKptStatus )
```

Discovery and Provisioning APIs for KPT

File: cpa_cy_kpt.h

Query KPT's issuing public key(R_Pu) and signature from QAT driver.

Description:

This function is to query the RSA3K issuing key and its PKCS#1 v2.2 SHA-384 signature from the QAT driver.

Context:

This function may sleep, and MUST NOT be called in interrupt context.

Assumptions:

None

Side-Effects:

None

Blocking:

This function is synchronous and blocking.

Parameters

in	*instanceHandle*	Instance handle.
out	*pIssueCert*	KPT-2.0 Issuing certificate in PEM format as defined in RFC#7468
out	*pKptStatus*	One of the status codes denoted in the enumerate type CpaCyKptKeyManagementStatus CPA_CY_KPT_SUCCESS Issuing key retrieved successfully CPA_CY_KPT_FAILED Operation failed

Return values

CPA_STATUS_SUCCESS	Function executed successfully.
CPA_STATUS_INVALID_PARAM	Invalid parameter passed in.
CPA_STATUS_FAIL	Function failed. Suggested course of action is to shutdown and restart.
CPA_STATUS_UNSUPPORTED	Function is not supported.
CPA_STATUS_RESTARTING	API implementation is restarting. Resubmit the request.

Precondition

The component has been initialized via cpaCyStartInstance function.

Postcondition

None

Note

Note that this is a synchronous function and has no completion callback associated with it.

See also

4.17.5.2 cpaCyKptQueryDeviceCredentials()

```
CpaStatus cpaCyKptQueryDeviceCredentials (
            const CpaInstanceHandle instanceHandle,
            CpaCyKptValidationKey * pDevCredential,
            CpaCyKptKeyManagementStatus * pKptStatus )
```

File: cpa_cy_kpt.h

Query KPT's Per-Part public key(I_pu) and signature from QAT device

Description:

This function is to query RSA3K Per-Part public key and its PKCS#1 v2.2 SHA-384 signature from the QAT device.

Context:

This function may sleep, and MUST NOT be called in interrupt context.

Assumptions:

None

Side-Effects:

None

Blocking:

This function is synchronous and blocking.

Parameters

in	*instanceHandle*	Instance handle.
out	*pDevCredential*	Device Per-Part public key
out	*pKptStatus*	One of the status codes denoted in the enumerate type CpaCyKptKeyManagementStatus CPA_CY_KPT_SUCCESS Device credentials retreived successfully CPA_CY_KPT_FAILED Operation failed

Return values

CPA_STATUS_SUCCESS	Function executed successfully.
CPA_STATUS_INVALID_PARAM	Invalid parameter passed in.
CPA_STATUS_FAIL	Function failed. Suggested course of action is to shutdown and restart.
CPA_STATUS_UNSUPPORTED	Function is not supported.
CPA_STATUS_RESTARTING	API implementation is restarting. Resubmit the request.

Precondition

The component has been initialized via cpaCyStartInstance function.

Postcondition

None

Note

Note that this is a synchronous function and has no completion callback associated with it.

See also

4.17.5.3 cpaCyKptLoadKey()

```
CpaStatus cpaCyKptLoadKey (
            CpaInstanceHandle instanceHandle,
            CpaCyKptLoadKey * pSWK,
            CpaCyKptHandle * keyHandle,
            CpaCyKptKeyManagementStatus * pKptStatus )
```

File: cpa_cy_kpt.h

Perform KPT key loading function.

Description:

This function is invoked by a QAT application to load an encrypted symmetric wrapping key.

Context:

This is a synchronous function and it can sleep. It MUST NOT be executed in a context that DOES NOT permit sleeping.

Assumptions:

None

Side-Effects:

None

Blocking:

This function is synchronous and blocking.

Reentrant:

No

Thread-safe:

Yes

Parameters

in	*instanceHandle*	QAT service instance handle.
in	*pSWK*	Encrypted SWK
out	*keyHandle*	A 64-bit handle value created by KPT
out	*pKptStatus*	One of the status codes denoted in the enumerate type CpaCyKptKeyManagementStatus CPA_CY_KPT_SUCCESS Key Loaded successfully CPA_CY_KPT_LOADKEY_FAIL_QUOTA_EXCEEDED_PER_VFID SWK count exceeds the configured maxmium value per VFID CPA_CY_KPT_LOADKEY_FAIL_QUOTA_EXCEEDED_PER_PASID SWK count exceeds the configured maxmium value per PASID CPA_CY_KPT_LOADKEY_FAIL_QUOTA_EXCEEDED SWK count exceeds the configured maxmium value when not scoped to VFID or PASID CPA_CY_KPT_FAILED Operation failed due to unspecified reason

Return values

CPA_STATUS_SUCCESS	Function executed successfully.
CPA_STATUS_FAIL	Function failed.
CPA_STATUS_INVALID_PARAM	Invalid parameter passed in.
CPA_STATUS_RESOURCE	Error related to system resources.
CPA_STATUS_RESTARTING	API implementation is restarting. Resubmit the request.
CPA_STATUS_UNSUPPORTED	KPT-2.0 is not supported.

Precondition

Component has been initialized.

Postcondition

None

Note

None

See also

None

4.17.5.4 cpaCyKptDeleteKey()

```
CpaStatus cpaCyKptDeleteKey (
            CpaInstanceHandle instanceHandle,
            CpaCyKptHandle keyHandle,
            CpaCyKptKeyManagementStatus * pKptStatus )
```

File: cpa_cy_kpt.h

Perform KPT delete keys function according to key handle

Description:

Before closing a QAT session(instance), an application that has previously stored its wrapping key in a QAT device using the KPT framework executes this call to delete its wrapping key in the QAT device.

Context:

This is a synchronous function and it can sleep. It MUST NOT be executed in a context that DOES NOT permit sleeping.

Assumptions:

None

Side-Effects:

None

Blocking:

This function is synchronous and blocking.

Reentrant:

No

Thread-safe:

Yes

Parameters

in	*instanceHandle*	QAT service instance handle.
in	*keyHandle*	A 64-bit handle value
out	*pkptstatus*	One of the status codes denoted in the enumerate type CpaCyKptKeyManagementStatus CPA_CY_KPT_SUCCESS Key Deleted successfully CPA_CY_KPT_SWK_FAIL_NOT_FOUND For any reason the input handle cannot be found. CPA_CY_KPT_FAILED Operation failed due to unspecified reason

Return values

CPA_STATUS_SUCCESS	Function executed successfully.
CPA_STATUS_FAIL	Function failed.

Return values

CPA_STATUS_INVALID_PARAM	Invalid parameter passed in.
CPA_STATUS_RESOURCE	Error related to system resources.
CPA_STATUS_RESTARTING	API implementation is restarting. Resubmit the request.

Precondition

Component has been initialized.

Postcondition

None

Note

None

See also

None

4.17.5.5 cpaCyKptRsaDecrypt()

```
CpaStatus cpaCyKptRsaDecrypt (
            const CpaInstanceHandle instanceHandle,
            const CpaCyGenFlatBufCbFunc pRsaDecryptCb,
            void * pCallbackTag,
            const CpaCyKptRsaDecryptOpData * pDecryptOpData,
            CpaFlatBuffer * pOutputData,
            CpaCyKptUnwrapContext * pKptUnwrapContext )
```

Usage APIs for KPT

File: cpa_cy_kpt.h

Perform KPT-2.0 mode RSA decrypt primitive operation on the input data.

Description:

This function is a variant of cpaCyRsaDecrypt, which will perform an RSA decryption primitive operation on the input data using the specified RSA private key which are encrypted. As the RSA decryption primitive and signing primitive operations are mathematically identical this function may also be used to perform an RSA signing primitive operation.

Context:

When called as an asynchronous function it cannot sleep. It can be executed in a context that does not permit sleeping. When called as a synchronous function it may sleep. It MUST NOT be executed in a context that DOES NOT permit sleeping.

Assumptions:

None

Side-Effects:

None

Blocking:

Yes when configured to operate in synchronous mode.

Reentrant:

No

Thread-safe:

Yes

Parameters

in	*instanceHandle*	Instance handle.
in	*pRsaDecryptCb*	Pointer to callback function to be invoked when the operation is complete. If this is set to a NULL value the function will operate synchronously.
in	*pCallbackTag*	Opaque User Data for this specific call. Will be returned unchanged in the callback.
in	*pDecryptOpData*	Structure containing all the data needed to perform the RSA decrypt operation. The client code allocates the memory for this structure. This component takes ownership of the memory until it is returned in the callback.
out	*pOutputData*	Pointer to structure into which the result of the RSA decryption primitive is written. The client MUST allocate this memory. The data pointed to is an integer in big-endian order. The value will be between 0 and the modulus n - 1. On invocation the callback function will contain this parameter in the pOut parameter.
in	*pKptUnwrapContext*	Pointer of structure into which the content of KptUnwrapContext is kept. The client MUST allocate this memory and copy structure KptUnwrapContext into this flat buffer.

Return values

CPA_STATUS_SUCCESS	Function executed successfully.
CPA_STATUS_FAIL	Function failed.
CPA_STATUS_RETRY	Resubmit the request.
CPA_STATUS_INVALID_PARAM	Invalid parameter passed in.

Return values

CPA_STATUS_RESOURCE	Error related to system resources.
CPA_STATUS_RESTARTING	API implementation is restarting.Resubmit the request.

Precondition

The component has been initialized via cpaCyStartInstance function.

Postcondition

None

Note

By virtue of invoking cpaSyKptRsaDecrypt, the implementation understands that pDecryptOpData contains an encrypted private key that requires unwrapping. KptUnwrapContext contains a 'KptHandle' field that points to the unwrapping key in the WKT. When pRsaDecryptCb is non-NULL an asynchronous callback is generated in response to this function call. Any errors generated during processing are reported as part of the callback status code. For optimal performance, data pointers SHOULD be 8-byte aligned. In KPT release, private key field in CpaCyKptRsaDecryptOpData is a concatenation of cipher text and hash tag. For optimal performance, data pointers SHOULD be 8-byte aligned.

See also

CpaCyKptRsaDecryptOpData, CpaCyGenFlatBufCbFunc,

4.17.5.6 cpaCyKptEcdsaSignRS()

```
CpaStatus cpaCyKptEcdsaSignRS (
            const CpaInstanceHandle instanceHandle,
            const CpaCyEcdsaSignRSCbFunc pCb,
            void * pCallbackTag,
            const CpaCyKptEcdsaSignRSOpData * pOpData,
            CpaBoolean * pSignStatus,
            CpaFlatBuffer * pR,
            CpaFlatBuffer * pS,
            CpaCyKptUnwrapContext * pKptUnwrapContext )
```

Generate ECDSA Signature R & S.

Description:

This function is a variant of cpaCyEcdsaSignRS, it generates ECDSA signature R & S as per ANSI X9.62 2005 section 7.3.

Context:

When called as an asynchronous function it cannot sleep. It can be executed in a context that does not permit sleeping. When called as a synchronous function it may sleep. It MUST NOT be executed in a context that DOES NOT permit sleeping.

Assumptions:

None

Side-Effects:

None

Blocking:

Yes when configured to operate in synchronous mode.

Reentrant:

No

Thread-safe:

Yes

Parameters

in	*instanceHandle*	Instance handle.
in	*pCb*	Callback function pointer. If this is set to a NULL value the function will operate synchronously.
in	*pCallbackTag*	User-supplied value to help identify request.
in	*pOpData*	Structure containing all the data needed to perform the operation. The client code allocates the memory for this structure. This component takes ownership of the memory until it is returned in the callback.
out	*pSignStatus*	In synchronous mode, the multiply output is valid (CPA_TRUE) or the output is invalid (CPA_FALSE).
out	*pR*	ECDSA message signature r.
out	*pS*	ECDSA message signature s.
in	*pKptUnwrapContext*	Pointer of structure into which the content of KptUnwrapContext is kept,The client MUST allocate this memory and copy structure KptUnwrapContext into this flat buffer.

Return values

CPA_STATUS_SUCCESS	Function executed successfully.
CPA_STATUS_FAIL	Function failed.
CPA_STATUS_RETRY	Resubmit the request.
CPA_STATUS_INVALID_PARAM	Invalid parameter passed in.
CPA_STATUS_RESOURCE	Error related to system resources.
CPA_STATUS_RESTARTING	API implementation is restarting. Resubmit the request.
CPA_STATUS_UNSUPPORTED	Function is not supported.

Precondition

The component has been initialized via cpaCyStartInstance function.

Postcondition

None

Note

By virtue of invoking the cpaCyKptEcdsaSignRS, the implementation understands CpaCyEcdsaSignRSOpData contains an encrypted private key that requires unwrapping. KptUnwrapContext contains a 'KptHandle' field that points to the unwrapping key in the WKT. When pCb is non-NULL an asynchronous callback of type CpaCyEcdsaSignRSCbFunc generated in response to this function call. In KPT release, private key field in CpaCyEcdsaSignRSOpData is a concatenation of cipher text and hash tag.

See also

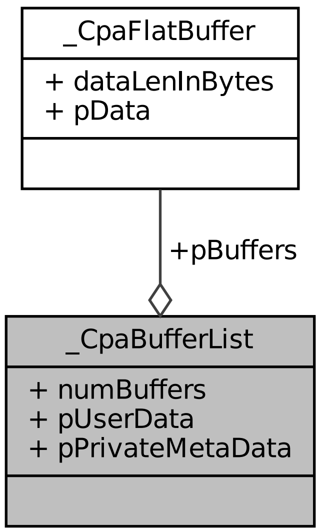
None

Chapter 5

Data Structure Documentation

5.1 _CpaBufferList Struct Reference

Collaboration diagram for _CpaBufferList:



Data Fields

- Cpa32U numBuffers
- CpaFlatBuffer ∗ pBuffers
- void ∗ pUserData
- void ∗ pPrivateMetaData

5.1.1 Detailed Description

Scatter/Gather buffer list containing an array of flat buffers.

Description:

A scatter/gather buffer list structure. This buffer structure is typically used to represent a region of memory which is not physically contiguous, by describing it as a collection of buffers, each of which is physically contiguous.

Note

The memory for the pPrivateMetaData member must be allocated by the client as physically contiguous memory. When allocating memory for pPrivateMetaData, a call to the corresponding BufferListGetMetaSize function (e.g. cpaCyBufferListGetMetaSize) MUST be made to determine the size of the Meta Data Buffer. The returned size (in bytes) may then be passed in a memory allocation routine to allocate the pPrivateMetaData memory.

Definition at line 164 of file cpa.h.

5.1.2 Field Documentation

5.1.2.1 numBuffers

`Cpa32U _CpaBufferList::numBuffers`

Number of buffers in the list

Definition at line 165 of file cpa.h.

5.1.2.2 pBuffers

`CpaFlatBuffer* _CpaBufferList::pBuffers`

Pointer to an unbounded array containing the number of CpaFlatBuffers defined by numBuffers

Definition at line 167 of file cpa.h.

5.1.2.3 pUserData

`void* _CpaBufferList::pUserData`

This is an opaque field that is not read or modified internally.

Definition at line 171 of file cpa.h.

5.1.2.4 pPrivateMetaData

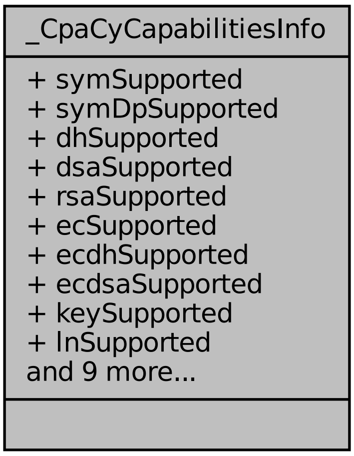
```
void* _CpaBufferList::pPrivateMetaData
```

Private representation of this buffer list. The memory for this buffer needs to be allocated by the client as contiguous data. The amount of memory required is returned with a call to the corresponding BufferListGetMetaSize function. If that function returns a size of zero then no memory needs to be allocated, and this parameter can be NULL.

Definition at line 173 of file cpa.h.

5.2 _CpaCyCapabilitiesInfo Struct Reference

Collaboration diagram for _CpaCyCapabilitiesInfo:



Data Fields

- CpaBoolean symSupported
- CpaBoolean symDpSupported
- CpaBoolean dhSupported
- CpaBoolean dsaSupported
- CpaBoolean rsaSupported
- CpaBoolean ecSupported
- CpaBoolean ecdhSupported
- CpaBoolean ecdsaSupported
- CpaBoolean keySupported
- CpaBoolean lnSupported
- CpaBoolean primeSupported
- CpaBoolean drbgSupported
- CpaBoolean nrbgSupported

- CpaBoolean randSupported
- CpaBoolean kptSupported
- CpaBoolean hkdfSupported
- CpaBoolean extAlgchainSupported
- CpaBoolean ecEdMontSupported
- CpaBoolean ecSm2Supported

5.2.1 Detailed Description

Cryptographic Capabilities Info

Description:

This structure contains the capabilities that vary across API implementations. This structure is used in conjunction with cpaCyQueryCapabilities() to determine the capabilities supported by a particular API implementation.

The client MUST allocate memory for this structure and any members that require memory. When the structure is passed into the function ownership of the memory passes to the function. Ownership of the memory returns to the client when the function returns.

Definition at line 156 of file cpa_cy_im.h.

5.2.2 Field Documentation

5.2.2.1 symSupported

`CpaBoolean _CpaCyCapabilitiesInfo::symSupported`

CPA_TRUE if instance supports the symmetric cryptography API. See
Symmetric Cipher and Hash Cryptographic API.

Definition at line 158 of file cpa_cy_im.h.

5.2.2.2 symDpSupported

`CpaBoolean _CpaCyCapabilitiesInfo::symDpSupported`

CPA_TRUE if instance supports the symmetric cryptography data plane API. See
Symmetric cryptographic Data Plane API.

Definition at line 161 of file cpa_cy_im.h.

5.2.2.3 dhSupported

`CpaBoolean _CpaCyCapabilitiesInfo::dhSupported`

CPA_TRUE if instance supports the Diffie Hellman API. See Diffie-Hellman (DH) API.

Definition at line 165 of file cpa_cy_im.h.

5.2.2.4 dsaSupported

`CpaBoolean _CpaCyCapabilitiesInfo::dsaSupported`

CPA_TRUE if instance supports the DSA API. See Digital Signature Algorithm (DSA) API.

Definition at line 168 of file cpa_cy_im.h.

5.2.2.5 rsaSupported

`CpaBoolean _CpaCyCapabilitiesInfo::rsaSupported`

CPA_TRUE if instance supports the RSA API. See RSA API.

Definition at line 171 of file cpa_cy_im.h.

5.2.2.6 ecSupported

`CpaBoolean _CpaCyCapabilitiesInfo::ecSupported`

CPA_TRUE if instance supports the Elliptic Curve API. See Elliptic Curve (EC) API.

Definition at line 174 of file cpa_cy_im.h.

5.2.2.7 ecdhSupported

`CpaBoolean _CpaCyCapabilitiesInfo::ecdhSupported`

CPA_TRUE if instance supports the Elliptic Curve Diffie Hellman API. See cpaCyEcdh.

Definition at line 177 of file cpa_cy_im.h.

5.2.2.8 ecdsaSupported

`CpaBoolean _CpaCyCapabilitiesInfo::ecdsaSupported`

CPA_TRUE if instance supports the Elliptic Curve DSA API. See Elliptic Curve Digital Signature Algorithm (ECDSA) API.

Definition at line 180 of file cpa_cy_im.h.

5.2.2.9 keySupported

`CpaBoolean _CpaCyCapabilitiesInfo::keySupported`

CPA_TRUE if instance supports the Key Generation API. See Cryptographic Key and Mask Generation API.

Definition at line 183 of file cpa_cy_im.h.

5.2.2.10 lnSupported

`CpaBoolean _CpaCyCapabilitiesInfo::lnSupported`

CPA_TRUE if instance supports the Large Number API. See Cryptographic Large Number API.

Definition at line 186 of file cpa_cy_im.h.

5.2.2.11 primeSupported

`CpaBoolean _CpaCyCapabilitiesInfo::primeSupported`

CPA_TRUE if instance supports the prime number testing API. See Prime Number Test API.

Definition at line 189 of file cpa_cy_im.h.

5.2.2.12 drbgSupported

`CpaBoolean _CpaCyCapabilitiesInfo::drbgSupported`

CPA_TRUE if instance supports the DRBG API. See cpaCyDrbg.

Definition at line 192 of file cpa_cy_im.h.

5.2.2.13 nrbgSupported

`CpaBoolean _CpaCyCapabilitiesInfo::nrbgSupported`

CPA_TRUE if instance supports the NRBG API. See cpaCyNrbg.

Definition at line 195 of file cpa_cy_im.h.

5.2.2.14 randSupported

`CpaBoolean _CpaCyCapabilitiesInfo::randSupported`

CPA_TRUE if instance supports the random bit/number generation API. See cpaCyRand.

Definition at line 198 of file cpa_cy_im.h.

5.2.2.15 kptSupported

`CpaBoolean _CpaCyCapabilitiesInfo::kptSupported`

CPA_TRUE if instance supports the Intel(R) KPT Cryptographic API. See Intel(R) Key Protection Technology (KPT) Cryptographic API.

Definition at line 201 of file cpa_cy_im.h.

5.2.2.16 hkdfSupported

`CpaBoolean _CpaCyCapabilitiesInfo::hkdfSupported`

CPA_TRUE if instance supports the HKDF components of the KeyGen API. See Cryptographic Key and Mask Generation API.

Definition at line 204 of file cpa_cy_im.h.

5.2.2.17 extAlgchainSupported

`CpaBoolean _CpaCyCapabilitiesInfo::extAlgchainSupported`

CPA_TRUE if instance supports algorithm chaining for certain wireless algorithms. Please refer to implementation for details. See Symmetric Cipher and Hash Cryptographic API.

Definition at line 207 of file cpa_cy_im.h.

5.2.2.18 ecEdMontSupported

```
CpaBoolean _CpaCyCapabilitiesInfo::ecEdMontSupported
```

CPA_TRUE if instance supports the Edwards and Montgomery elliptic curves of the EC API. See Elliptic Curve (EC) API

Definition at line 211 of file cpa_cy_im.h.

5.2.2.19 ecSm2Supported

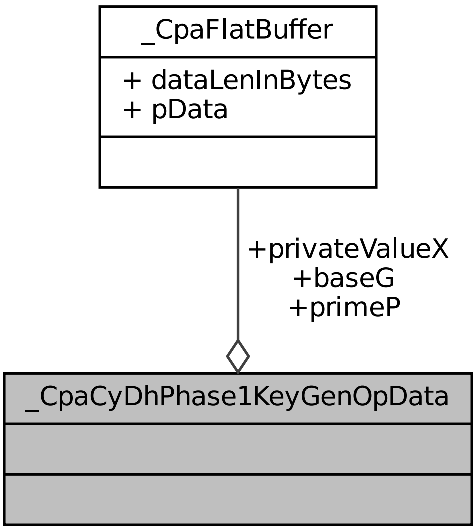
```
CpaBoolean _CpaCyCapabilitiesInfo::ecSm2Supported
```

CPA_TRUE if instance supports the EcSM2 API. See cpaCyEcsm2.

Definition at line 215 of file cpa_cy_im.h.

5.3 _CpaCyDhPhase1KeyGenOpData Struct Reference

Collaboration diagram for _CpaCyDhPhase1KeyGenOpData:



Data Fields

- CpaFlatBuffer primeP
- CpaFlatBuffer baseG
- CpaFlatBuffer privateValueX

5.3.1 Detailed Description

Diffie-Hellman Phase 1 Key Generation Data.

Description:

This structure lists the different items that are required in the cpaCyDhKeyGenPhase1 function. The client MUST allocate the memory for this structure. When the structure is passed into the function, ownership of the memory passes to the function. Ownership of the memory returns to the client when this structure is returned with the CpaCyDhPhase1KeyGenOpData structure.

Note

If the client modifies or frees the memory referenced in this structure after it has been submitted to the cpaCyDhKeyGenPhase1 function, and before it has been returned in the callback, undefined behavior will result. All values in this structure are required to be in Most Significant Byte first order, e.g. primeP.pData[0] = MSB.

Definition at line 58 of file cpa_cy_dh.h.

5.3.2 Field Documentation

5.3.2.1 primeP

`CpaFlatBuffer _CpaCyDhPhase1KeyGenOpData::primeP`

Flat buffer containing a pointer to the random odd prime number (p). The bit-length of this number may be one of 768, 1024, 1536, 2048, 3072, 4096 or 8192.

Definition at line 59 of file cpa_cy_dh.h.

5.3.2.2 baseG

`CpaFlatBuffer _CpaCyDhPhase1KeyGenOpData::baseG`

Flat buffer containing a pointer to base (g). This MUST comply with the following: $0 < g < p$.

Definition at line 64 of file cpa_cy_dh.h.

5.3.2.3 privateValueX

```
CpaFlatBuffer _CpaCyDhPhase1KeyGenOpData::privateValueX
```

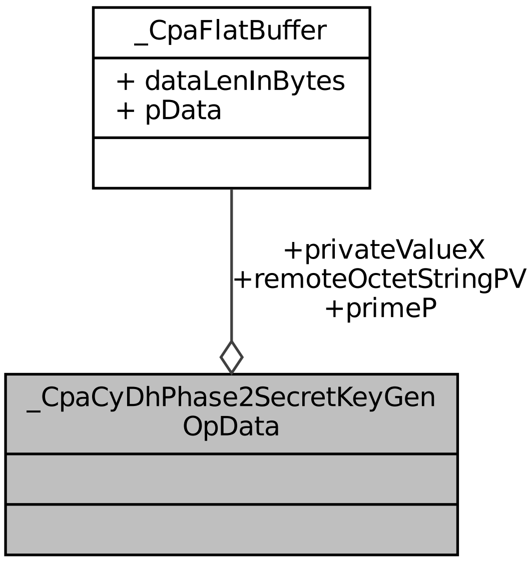
Flat buffer containing a pointer to the private value (x). This is a random value which MUST satisfy the following condition: $0 <$ PrivateValueX $<$ (PrimeP - 1)

Refer to PKCS #3: Diffie-Hellman Key-Agreement Standard for details. The client creating this data MUST ensure the compliance of this value with the standard. Note: This value is also needed to complete local phase 2 Diffie-Hellman operation.

Definition at line 69 of file cpa_cy_dh.h.

5.4 _CpaCyDhPhase2SecretKeyGenOpData Struct Reference

Collaboration diagram for _CpaCyDhPhase2SecretKeyGenOpData:



Data Fields

- CpaFlatBuffer primeP
- CpaFlatBuffer remoteOctetStringPV
- CpaFlatBuffer privateValueX

5.4.1 Detailed Description

Diffie-Hellman Phase 2 Secret Key Generation Data.

Description:

This structure lists the different items that required in the cpaCyDhKeyGenPhase2Secret function. The client MUST allocate the memory for this structure. When the structure is passed into the function, ownership of the memory passes to the function. Ownership of the memory returns to the client when this structure is returned with the callback.

Note

If the client modifies or frees the memory referenced in this structure after it has been submitted to the cpaCyDhKeyGenPhase2Secret function, and before it has been returned in the callback, undefined behavior will result. All values in this structure are required to be in Most Significant Byte first order, e.g. primeP.pData[0] = MSB.

Definition at line 100 of file cpa_cy_dh.h.

5.4.2 Field Documentation

5.4.2.1 primeP

`CpaFlatBuffer _CpaCyDhPhase2SecretKeyGenOpData::primeP`

Flat buffer containing a pointer to the random odd prime number (p). The bit-length of this number may be one of 768, 1024, 1536, 2048, 3072, 4096 or 8192. This SHOULD be same prime number as was used in the phase 1 key generation operation.

Definition at line 101 of file cpa_cy_dh.h.

5.4.2.2 remoteOctetStringPV

`CpaFlatBuffer _CpaCyDhPhase2SecretKeyGenOpData::remoteOctetStringPV`

Flat buffer containing a pointer to the remote entity octet string Public Value (PV).

Definition at line 107 of file cpa_cy_dh.h.

5.4.2.3 privateValueX

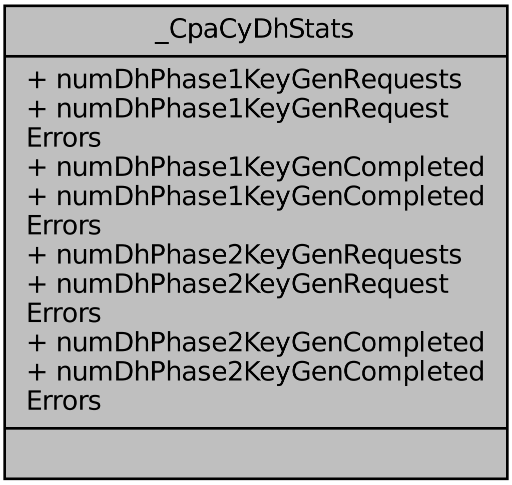
`CpaFlatBuffer _CpaCyDhPhase2SecretKeyGenOpData::privateValueX`

Flat buffer containing a pointer to the private value (x). This value may have been used in a call to the cpaCyDhKeyGenPhase1 function. This is a random value which MUST satisfy the following condition: 0 < privateValueX < (primeP - 1).

Definition at line 110 of file cpa_cy_dh.h.

5.5 _CpaCyDhStats Struct Reference

Collaboration diagram for _CpaCyDhStats:



Data Fields

- Cpa32U numDhPhase1KeyGenRequests
- Cpa32U numDhPhase1KeyGenRequestErrors
- Cpa32U numDhPhase1KeyGenCompleted
- Cpa32U numDhPhase1KeyGenCompletedErrors
- Cpa32U numDhPhase2KeyGenRequests
- Cpa32U numDhPhase2KeyGenRequestErrors
- Cpa32U numDhPhase2KeyGenCompleted
- Cpa32U numDhPhase2KeyGenCompletedErrors

5.5.1 Detailed Description

Diffie-Hellman Statistics.

Deprecated As of v1.3 of the Crypto API, this structure has been deprecated, replaced by CpaCyDhStats64.

Description:

This structure contains statistics on the Diffie-Hellman operations. Statistics are set to zero when the component is initialized, and are collected per instance.

Definition at line 129 of file cpa_cy_dh.h.

5.5.2 Field Documentation

5.5.2.1 numDhPhase1KeyGenRequests

`Cpa32U _CpaCyDhStats::numDhPhase1KeyGenRequests`

Total number of successful Diffie-Hellman phase 1 key generation requests.

Definition at line 130 of file cpa_cy_dh.h.

5.5.2.2 numDhPhase1KeyGenRequestErrors

`Cpa32U _CpaCyDhStats::numDhPhase1KeyGenRequestErrors`

Total number of Diffie-Hellman phase 1 key generation requests that had an error and could not be processed.

Definition at line 133 of file cpa_cy_dh.h.

5.5.2.3 numDhPhase1KeyGenCompleted

`Cpa32U _CpaCyDhStats::numDhPhase1KeyGenCompleted`

Total number of Diffie-Hellman phase 1 key generation operations that completed successfully.

Definition at line 136 of file cpa_cy_dh.h.

5.5.2.4 numDhPhase1KeyGenCompletedErrors

`Cpa32U _CpaCyDhStats::numDhPhase1KeyGenCompletedErrors`

Total number of Diffie-Hellman phase 1 key generation operations that could not be completed successfully due to errors.

Definition at line 139 of file cpa_cy_dh.h.

5.5.2.5 numDhPhase2KeyGenRequests

`Cpa32U _CpaCyDhStats::numDhPhase2KeyGenRequests`

Total number of successful Diffie-Hellman phase 2 key generation requests.

Definition at line 142 of file cpa_cy_dh.h.

5.5.2.6 numDhPhase2KeyGenRequestErrors

`Cpa32U _CpaCyDhStats::numDhPhase2KeyGenRequestErrors`

Total number of Diffie-Hellman phase 2 key generation requests that had an error and could not be processed.

Definition at line 145 of file cpa_cy_dh.h.

5.5.2.7 numDhPhase2KeyGenCompleted

`Cpa32U _CpaCyDhStats::numDhPhase2KeyGenCompleted`

Total number of Diffie-Hellman phase 2 key generation operations that completed successfully.

Definition at line 148 of file cpa_cy_dh.h.

5.5.2.8 numDhPhase2KeyGenCompletedErrors

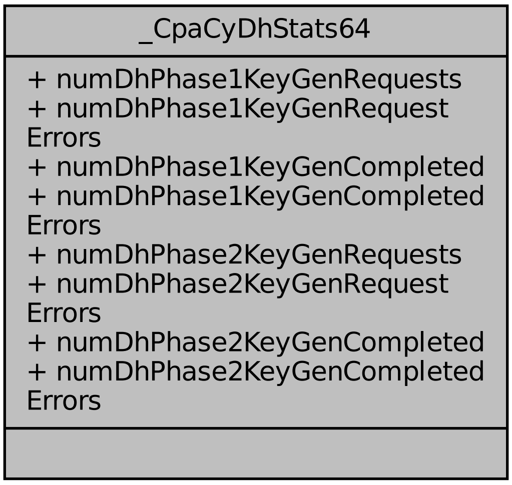
`Cpa32U _CpaCyDhStats::numDhPhase2KeyGenCompletedErrors`

Total number of Diffie-Hellman phase 2 key generation operations that could not be completed successfully due to errors.

Definition at line 151 of file cpa_cy_dh.h.

5.6 _CpaCyDhStats64 Struct Reference

Collaboration diagram for _CpaCyDhStats64:



Data Fields

- Cpa64U numDhPhase1KeyGenRequests
- Cpa64U numDhPhase1KeyGenRequestErrors
- Cpa64U numDhPhase1KeyGenCompleted
- Cpa64U numDhPhase1KeyGenCompletedErrors
- Cpa64U numDhPhase2KeyGenRequests
- Cpa64U numDhPhase2KeyGenRequestErrors
- Cpa64U numDhPhase2KeyGenCompleted
- Cpa64U numDhPhase2KeyGenCompletedErrors

5.6.1 Detailed Description

Diffie-Hellman Statistics (64-bit version).

Description:

This structure contains the 64-bit version of the statistics on the Diffie-Hellman operations. Statistics are set to zero when the component is initialized, and are collected per instance.

Definition at line 166 of file cpa_cy_dh.h.

5.6.2 Field Documentation

5.6.2.1 numDhPhase1KeyGenRequests

`Cpa64U _CpaCyDhStats64::numDhPhase1KeyGenRequests`

Total number of successful Diffie-Hellman phase 1 key generation requests.

Definition at line 167 of file cpa_cy_dh.h.

5.6.2.2 numDhPhase1KeyGenRequestErrors

`Cpa64U _CpaCyDhStats64::numDhPhase1KeyGenRequestErrors`

Total number of Diffie-Hellman phase 1 key generation requests that had an error and could not be processed.

Definition at line 170 of file cpa_cy_dh.h.

5.6.2.3 numDhPhase1KeyGenCompleted

`Cpa64U _CpaCyDhStats64::numDhPhase1KeyGenCompleted`

Total number of Diffie-Hellman phase 1 key generation operations that completed successfully.

Definition at line 173 of file cpa_cy_dh.h.

5.6.2.4 numDhPhase1KeyGenCompletedErrors

`Cpa64U _CpaCyDhStats64::numDhPhase1KeyGenCompletedErrors`

Total number of Diffie-Hellman phase 1 key generation operations that could not be completed successfully due to errors.

Definition at line 176 of file cpa_cy_dh.h.

5.6.2.5 numDhPhase2KeyGenRequests

`Cpa64U _CpaCyDhStats64::numDhPhase2KeyGenRequests`

Total number of successful Diffie-Hellman phase 2 key generation requests.

Definition at line 179 of file cpa_cy_dh.h.

5.6.2.6 numDhPhase2KeyGenRequestErrors

`Cpa64U _CpaCyDhStats64::numDhPhase2KeyGenRequestErrors`

Total number of Diffie-Hellman phase 2 key generation requests that had an error and could not be processed.

Definition at line 182 of file cpa_cy_dh.h.

5.6.2.7 numDhPhase2KeyGenCompleted

`Cpa64U _CpaCyDhStats64::numDhPhase2KeyGenCompleted`

Total number of Diffie-Hellman phase 2 key generation operations that completed successfully.

Definition at line 185 of file cpa_cy_dh.h.

5.6.2.8 numDhPhase2KeyGenCompletedErrors

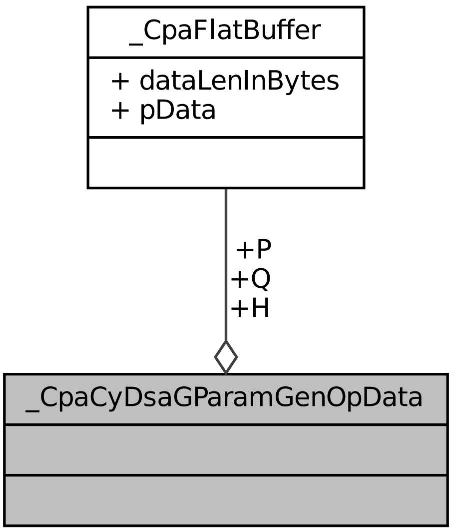
`Cpa64U _CpaCyDhStats64::numDhPhase2KeyGenCompletedErrors`

Total number of Diffie-Hellman phase 2 key generation operations that could not be completed successfully due to errors.

Definition at line 188 of file cpa_cy_dh.h.

5.7 _CpaCyDsaGParamGenOpData Struct Reference

Collaboration diagram for _CpaCyDsaGParamGenOpData:



Data Fields

- CpaFlatBuffer P
- CpaFlatBuffer Q
- CpaFlatBuffer H

5.7.1 Detailed Description

DSA G Parameter Generation Operation Data.

Description:

This structure contains the operation data for the cpaCyDsaGenGParam function. The client MUST allocate the memory for this structure and the items pointed to by this structure. When the structure is passed into the function, ownership of the memory passes to the function. Ownership of the memory returns to the client when this structure is returned in the callback function.

All values in this structure are required to be in Most Significant Byte first order, e.g. P.pData[0] = MSB.

All numbers MUST be stored in big-endian order.

Note

If the client modifies or frees the memory referenced in this structure after it has been submitted to the cpaCyDsaGenGParam function, and before it has been returned in the callback, undefined behavior will result.

See also

cpaCyDsaGenGParam()

Definition at line 124 of file cpa_cy_dsa.h.

5.7.2 Field Documentation

5.7.2.1 P

`CpaFlatBuffer _CpaCyDsaGParamGenOpData::P`

DSA group parameter p

Definition at line 125 of file cpa_cy_dsa.h.

5.7.2.2 Q

`CpaFlatBuffer _CpaCyDsaGParamGenOpData::Q`

DSA group parameter q

Definition at line 127 of file cpa_cy_dsa.h.

5.7.2.3 H

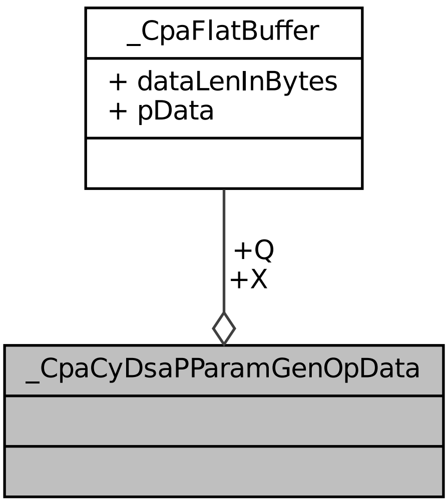
`CpaFlatBuffer _CpaCyDsaGParamGenOpData::H`

any integer with 1 < h < p - 1

Definition at line 129 of file cpa_cy_dsa.h.

5.8 _CpaCyDsaPParamGenOpData Struct Reference

Collaboration diagram for _CpaCyDsaPParamGenOpData:



Data Fields

- CpaFlatBuffer X
- CpaFlatBuffer Q

5.8.1 Detailed Description

DSA P Parameter Generation Operation Data.

Description:

This structure contains the operation data for the cpaCyDsaGenPParam function. The client MUST allocate the memory for this structure and the items pointed to by this structure. When the structure is passed into the function, ownership of the memory passes to the function. Ownership of the memory returns to the client when this structure is returned in the callback function.

For optimal performance all data buffers SHOULD be 8-byte aligned.

All values in this structure are required to be in Most Significant Byte first order, e.g. X.pData[0] = MSB.

Note

If the client modifies or frees the memory referenced in this structure after it has been submitted to the cpaCyDsaGenPParam function, and before it has been returned in the callback, undefined behavior will result.

See also

cpaCyDsaGenPParam()

Definition at line 90 of file cpa_cy_dsa.h.

5.8.2 Field Documentation

5.8.2.1 X

`CpaFlatBuffer _CpaCyDsaPParamGenOpData::X`

$2^{(L-1)} <= X < 2^L$ (from FIPS 186-3)

Definition at line 91 of file cpa_cy_dsa.h.

5.8.2.2 Q

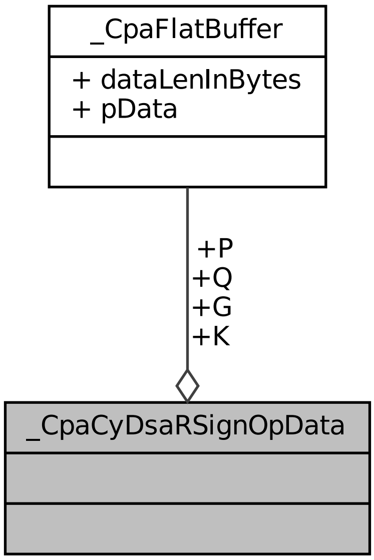
`CpaFlatBuffer _CpaCyDsaPParamGenOpData::Q`

DSA group parameter q

Definition at line 93 of file cpa_cy_dsa.h.

5.9 _CpaCyDsaRSignOpData Struct Reference

Collaboration diagram for _CpaCyDsaRSignOpData:



Data Fields

- CpaFlatBuffer P
- CpaFlatBuffer Q
- CpaFlatBuffer G
- CpaFlatBuffer K

5.9.1 Detailed Description

DSA R Sign Operation Data.

Description:

This structure contains the operation data for the cpaCyDsaSignR function. The client MUST allocate the memory for this structure and the items pointed to by this structure. When the structure is passed into the function, ownership of the memory passes to the function. Ownership of the memory returns to the client when this structure is returned in the callback function.

For optimal performance all data SHOULD be 8-byte aligned.

All values in this structure are required to be in Most Significant Byte first order, e.g. P.pData[0] = MSB.

Note

If the client modifies or frees the memory referenced in this structure after it has been submitted to the cpaCyDsaSignR function, and before it has been returned in the callback, undefined behavior will result.

See also

cpaCyDsaSignR()

Definition at line 196 of file cpa_cy_dsa.h.

5.9.2 Field Documentation

5.9.2.1 P

`CpaFlatBuffer _CpaCyDsaRSignOpData::P`

DSA group parameter p

Definition at line 197 of file cpa_cy_dsa.h.

5.9.2.2 Q

`CpaFlatBuffer _CpaCyDsaRSignOpData::Q`

DSA group parameter q

Definition at line 199 of file cpa_cy_dsa.h.

5.9.2.3 G

`CpaFlatBuffer _CpaCyDsaRSignOpData::G`

DSA group parameter g

Definition at line 201 of file cpa_cy_dsa.h.

5.9.2.4 K

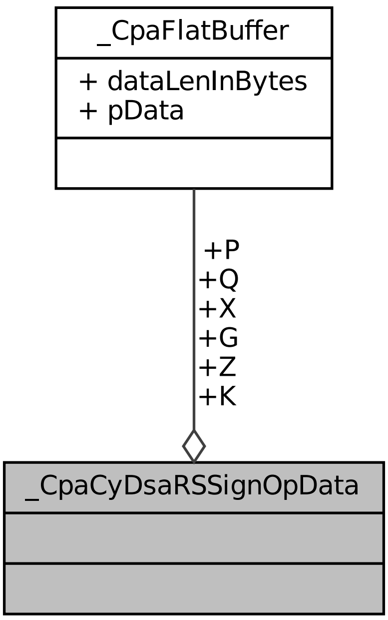
`CpaFlatBuffer _CpaCyDsaRSignOpData::K`

DSA secret parameter k for signing

Definition at line 203 of file cpa_cy_dsa.h.

5.10 _CpaCyDsaRSSignOpData Struct Reference

Collaboration diagram for _CpaCyDsaRSSignOpData:



Data Fields

- CpaFlatBuffer P
- CpaFlatBuffer Q
- CpaFlatBuffer G
- CpaFlatBuffer X
- CpaFlatBuffer K
- CpaFlatBuffer Z

5.10.1 Detailed Description

DSA R & S Sign Operation Data.

Description:

This structure contains the operation data for the cpaCyDsaSignRS function. The client MUST allocate the memory for this structure and the items pointed to by this structure. When the structure is passed into the function, ownership of the memory passes to the function. Ownership of the memory returns to the client when this structure is returned in the callback function.

For optimal performance all data SHOULD be 8-byte aligned.

All values in this structure are required to be in Most Significant Byte first order, e.g. P.pData[0] = MSB.

Note

If the client modifies or frees the memory referenced in this structure after it has been submitted to the cpaCyDsaSignRS function, and before it has been returned in the callback, undefined behavior will result.

See also

cpaCyDsaSignRS()

Definition at line 278 of file cpa_cy_dsa.h.

5.10.2 Field Documentation

5.10.2.1 P

`CpaFlatBuffer _CpaCyDsaRSSignOpData::P`

DSA group parameter p

Definition at line 279 of file cpa_cy_dsa.h.

5.10.2.2 Q

`CpaFlatBuffer _CpaCyDsaRSSignOpData::Q`

DSA group parameter q

Definition at line 281 of file cpa_cy_dsa.h.

5.10.2.3 G

`CpaFlatBuffer _CpaCyDsaRSSignOpData::G`

DSA group parameter g

Definition at line 283 of file cpa_cy_dsa.h.

5.10.2.4 X

`CpaFlatBuffer _CpaCyDsaRSSignOpData::X`

DSA private key x

Definition at line 285 of file cpa_cy_dsa.h.

5.10.2.5 K

`CpaFlatBuffer _CpaCyDsaRSSignOpData::K`

DSA secret parameter k for signing

Definition at line 287 of file cpa_cy_dsa.h.

5.10.2.6 Z

`CpaFlatBuffer _CpaCyDsaRSSignOpData::Z`

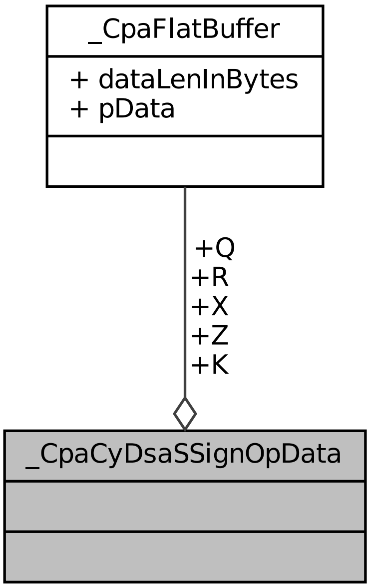
The leftmost min(N, outlen) bits of Hash(M), where:

- N is the bit length of q
- outlen is the bit length of the hash function output block
- M is the message to be signed

Definition at line 289 of file cpa_cy_dsa.h.

5.11 _CpaCyDsaSSignOpData Struct Reference

Collaboration diagram for _CpaCyDsaSSignOpData:



Data Fields

- CpaFlatBuffer Q
- CpaFlatBuffer X
- CpaFlatBuffer K
- CpaFlatBuffer R
- CpaFlatBuffer Z

5.11.1 Detailed Description

DSA S Sign Operation Data.

Description:

This structure contains the operation data for the cpaCyDsaSignS function. The client MUST allocate the memory for this structure and the items pointed to by this structure. When the structure is passed into the function, ownership of the memory passes to the function. Ownership of the memory returns to the client when this structure is returned in the callback function.

For optimal performance all data SHOULD be 8-byte aligned.

All values in this structure are required to be in Most Significant Byte first order, e.g. Q.pData[0] = MSB.

Note

If the client modifies or frees the memory referenced in this structure after it has been submitted to the cpaCyDsaSignS function, and before it has been returned in the callback, undefined behavior will result.

See also

cpaCyDsaSignS()

Definition at line 234 of file cpa_cy_dsa.h.

5.11.2 Field Documentation

5.11.2.1 Q

`CpaFlatBuffer _CpaCyDsaSSignOpData::Q`

DSA group parameter q

Definition at line 235 of file cpa_cy_dsa.h.

5.11.2.2 X

`CpaFlatBuffer _CpaCyDsaSSignOpData::X`

DSA private key x

Definition at line 237 of file cpa_cy_dsa.h.

5.11.2.3 K

`CpaFlatBuffer _CpaCyDsaSSignOpData::K`

DSA secret parameter k for signing

Definition at line 239 of file cpa_cy_dsa.h.

5.11.2.4 R

`CpaFlatBuffer _CpaCyDsaSSignOpData::R`

DSA message signature r

Definition at line 241 of file cpa_cy_dsa.h.

5.11.2.5 Z

`CpaFlatBuffer _CpaCyDsaSSignOpData::Z`

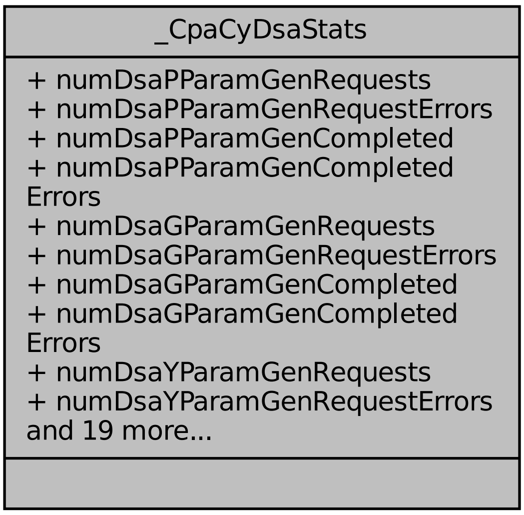
The leftmost min(N, outlen) bits of Hash(M), where:

- N is the bit length of q
- outlen is the bit length of the hash function output block
- M is the message to be signed

Definition at line 243 of file cpa_cy_dsa.h.

5.12 _CpaCyDsaStats Struct Reference

Collaboration diagram for _CpaCyDsaStats:



Data Fields

- Cpa32U numDsaPParamGenRequests
- Cpa32U numDsaPParamGenRequestErrors
- Cpa32U numDsaPParamGenCompleted
- Cpa32U numDsaPParamGenCompletedErrors
- Cpa32U numDsaGParamGenRequests
- Cpa32U numDsaGParamGenRequestErrors
- Cpa32U numDsaGParamGenCompleted
- Cpa32U numDsaGParamGenCompletedErrors
- Cpa32U numDsaYParamGenRequests
- Cpa32U numDsaYParamGenRequestErrors
- Cpa32U numDsaYParamGenCompleted
- Cpa32U numDsaYParamGenCompletedErrors
- Cpa32U numDsaRSignRequests
- Cpa32U numDsaRSignRequestErrors
- Cpa32U numDsaRSignCompleted
- Cpa32U numDsaRSignCompletedErrors
- Cpa32U numDsaSSignRequests
- Cpa32U numDsaSSignRequestErrors
- Cpa32U numDsaSSignCompleted
- Cpa32U numDsaSSignCompletedErrors
- Cpa32U numDsaRSSignRequests
- Cpa32U numDsaRSSignRequestErrors
- Cpa32U numDsaRSSignCompleted
- Cpa32U numDsaRSSignCompletedErrors
- Cpa32U numDsaVerifyRequests
- Cpa32U numDsaVerifyRequestErrors
- Cpa32U numDsaVerifyCompleted
- Cpa32U numDsaVerifyCompletedErrors
- Cpa32U numDsaVerifyFailures

5.12.1 Detailed Description

Cryptographic DSA Statistics.

Deprecated As of v1.3 of the Crypto API, this structure has been deprecated, replaced by CpaCyDsaStats64.

Description:

This structure contains statistics on the Cryptographic DSA operations. Statistics are set to zero when the component is initialized, and are collected per instance.

Definition at line 357 of file cpa_cy_dsa.h.

5.12.2 Field Documentation

5.12.2.1 numDsaPParamGenRequests

`Cpa32U _CpaCyDsaStats::numDsaPParamGenRequests`

Total number of successful DSA P parameter generation requests.

Definition at line 358 of file cpa_cy_dsa.h.

5.12.2.2 numDsaPParamGenRequestErrors

`Cpa32U _CpaCyDsaStats::numDsaPParamGenRequestErrors`

Total number of DSA P parameter generation requests that had an error and could not be processed.

Definition at line 360 of file cpa_cy_dsa.h.

5.12.2.3 numDsaPParamGenCompleted

`Cpa32U _CpaCyDsaStats::numDsaPParamGenCompleted`

Total number of DSA P parameter generation operations that completed successfully.

Definition at line 363 of file cpa_cy_dsa.h.

5.12.2.4 numDsaPParamGenCompletedErrors

`Cpa32U _CpaCyDsaStats::numDsaPParamGenCompletedErrors`

Total number of DSA P parameter generation operations that could not be completed successfully due to errors.

Definition at line 366 of file cpa_cy_dsa.h.

5.12.2.5 numDsaGParamGenRequests

`Cpa32U _CpaCyDsaStats::numDsaGParamGenRequests`

Total number of successful DSA G parameter generation requests.

Definition at line 369 of file cpa_cy_dsa.h.

5.12.2.6 numDsaGParamGenRequestErrors

`Cpa32U _CpaCyDsaStats::numDsaGParamGenRequestErrors`

Total number of DSA G parameter generation requests that had an error and could not be processed.

Definition at line 371 of file cpa_cy_dsa.h.

5.12.2.7 numDsaGParamGenCompleted

`Cpa32U _CpaCyDsaStats::numDsaGParamGenCompleted`

Total number of DSA G parameter generation operations that completed successfully.

Definition at line 374 of file cpa_cy_dsa.h.

5.12.2.8 numDsaGParamGenCompletedErrors

`Cpa32U _CpaCyDsaStats::numDsaGParamGenCompletedErrors`

Total number of DSA G parameter generation operations that could not be completed successfully due to errors.

Definition at line 377 of file cpa_cy_dsa.h.

5.12.2.9 numDsaYParamGenRequests

`Cpa32U _CpaCyDsaStats::numDsaYParamGenRequests`

Total number of successful DSA Y parameter generation requests.

Definition at line 380 of file cpa_cy_dsa.h.

5.12.2.10 numDsaYParamGenRequestErrors

`Cpa32U _CpaCyDsaStats::numDsaYParamGenRequestErrors`

Total number of DSA Y parameter generation requests that had an error and could not be processed.

Definition at line 382 of file cpa_cy_dsa.h.

5.12.2.11 numDsaYParamGenCompleted

`Cpa32U _CpaCyDsaStats::numDsaYParamGenCompleted`

Total number of DSA Y parameter generation operations that completed successfully.

Definition at line 385 of file cpa_cy_dsa.h.

5.12.2.12 numDsaYParamGenCompletedErrors

`Cpa32U _CpaCyDsaStats::numDsaYParamGenCompletedErrors`

Total number of DSA Y parameter generation operations that could not be completed successfully due to errors.

Definition at line 388 of file cpa_cy_dsa.h.

5.12.2.13 numDsaRSignRequests

`Cpa32U _CpaCyDsaStats::numDsaRSignRequests`

Total number of successful DSA R sign generation requests.

Definition at line 391 of file cpa_cy_dsa.h.

5.12.2.14 numDsaRSignRequestErrors

`Cpa32U _CpaCyDsaStats::numDsaRSignRequestErrors`

Total number of DSA R sign requests that had an error and could not be processed.

Definition at line 393 of file cpa_cy_dsa.h.

5.12.2.15 numDsaRSignCompleted

`Cpa32U _CpaCyDsaStats::numDsaRSignCompleted`

Total number of DSA R sign operations that completed successfully.

Definition at line 396 of file cpa_cy_dsa.h.

5.12.2.16 numDsaRSignCompletedErrors

`Cpa32U _CpaCyDsaStats::numDsaRSignCompletedErrors`

Total number of DSA R sign operations that could not be completed successfully due to errors.

Definition at line 399 of file cpa_cy_dsa.h.

5.12.2.17 numDsaSSignRequests

`Cpa32U _CpaCyDsaStats::numDsaSSignRequests`

Total number of successful DSA S sign generation requests.

Definition at line 402 of file cpa_cy_dsa.h.

5.12.2.18 numDsaSSignRequestErrors

`Cpa32U _CpaCyDsaStats::numDsaSSignRequestErrors`

Total number of DSA S sign requests that had an error and could not be processed.

Definition at line 404 of file cpa_cy_dsa.h.

5.12.2.19 numDsaSSignCompleted

`Cpa32U _CpaCyDsaStats::numDsaSSignCompleted`

Total number of DSA S sign operations that completed successfully.

Definition at line 407 of file cpa_cy_dsa.h.

5.12.2.20 numDsaSSignCompletedErrors

`Cpa32U _CpaCyDsaStats::numDsaSSignCompletedErrors`

Total number of DSA S sign operations that could not be completed successfully due to errors.

Definition at line 410 of file cpa_cy_dsa.h.

5.12.2.21 numDsaRSSignRequests

`Cpa32U _CpaCyDsaStats::numDsaRSSignRequests`

Total number of successful DSA RS sign generation requests.

Definition at line 413 of file cpa_cy_dsa.h.

5.12.2.22 numDsaRSSignRequestErrors

`Cpa32U _CpaCyDsaStats::numDsaRSSignRequestErrors`

Total number of DSA RS sign requests that had an error and could not be processed.

Definition at line 415 of file cpa_cy_dsa.h.

5.12.2.23 numDsaRSSignCompleted

`Cpa32U _CpaCyDsaStats::numDsaRSSignCompleted`

Total number of DSA RS sign operations that completed successfully.

Definition at line 418 of file cpa_cy_dsa.h.

5.12.2.24 numDsaRSSignCompletedErrors

`Cpa32U _CpaCyDsaStats::numDsaRSSignCompletedErrors`

Total number of DSA RS sign operations that could not be completed successfully due to errors.

Definition at line 421 of file cpa_cy_dsa.h.

5.12.2.25 numDsaVerifyRequests

`Cpa32U _CpaCyDsaStats::numDsaVerifyRequests`

Total number of successful DSA verify generation requests.

Definition at line 424 of file cpa_cy_dsa.h.

5.12.2.26 numDsaVerifyRequestErrors

`Cpa32U _CpaCyDsaStats::numDsaVerifyRequestErrors`

Total number of DSA verify requests that had an error and could not be processed.

Definition at line 426 of file cpa_cy_dsa.h.

5.12.2.27 numDsaVerifyCompleted

`Cpa32U _CpaCyDsaStats::numDsaVerifyCompleted`

Total number of DSA verify operations that completed successfully.

Definition at line 429 of file cpa_cy_dsa.h.

5.12.2.28 numDsaVerifyCompletedErrors

`Cpa32U _CpaCyDsaStats::numDsaVerifyCompletedErrors`

Total number of DSA verify operations that could not be completed successfully due to errors.

Definition at line 432 of file cpa_cy_dsa.h.

5.12.2.29 numDsaVerifyFailures

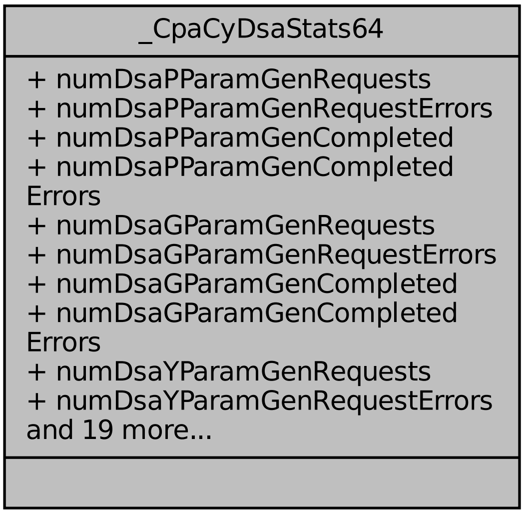
`Cpa32U _CpaCyDsaStats::numDsaVerifyFailures`

Total number of DSA verify operations that executed successfully but the outcome of the test was that the verification failed. Note that this does not indicate an error.

Definition at line 435 of file cpa_cy_dsa.h.

5.13 _CpaCyDsaStats64 Struct Reference

Collaboration diagram for _CpaCyDsaStats64:



Data Fields

- Cpa64U numDsaPParamGenRequests
- Cpa64U numDsaPParamGenRequestErrors
- Cpa64U numDsaPParamGenCompleted
- Cpa64U numDsaPParamGenCompletedErrors
- Cpa64U numDsaGParamGenRequests
- Cpa64U numDsaGParamGenRequestErrors
- Cpa64U numDsaGParamGenCompleted
- Cpa64U numDsaGParamGenCompletedErrors
- Cpa64U numDsaYParamGenRequests
- Cpa64U numDsaYParamGenRequestErrors
- Cpa64U numDsaYParamGenCompleted
- Cpa64U numDsaYParamGenCompletedErrors
- Cpa64U numDsaRSignRequests
- Cpa64U numDsaRSignRequestErrors
- Cpa64U numDsaRSignCompleted
- Cpa64U numDsaRSignCompletedErrors
- Cpa64U numDsaSSignRequests
- Cpa64U numDsaSSignRequestErrors
- Cpa64U numDsaSSignCompleted
- Cpa64U numDsaSSignCompletedErrors
- Cpa64U numDsaRSSignRequests
- Cpa64U numDsaRSSignRequestErrors
- Cpa64U numDsaRSSignCompleted
- Cpa64U numDsaRSSignCompletedErrors

- Cpa64U numDsaVerifyRequests
- Cpa64U numDsaVerifyRequestErrors
- Cpa64U numDsaVerifyCompleted
- Cpa64U numDsaVerifyCompletedErrors
- Cpa64U numDsaVerifyFailures

5.13.1 Detailed Description

Cryptographic DSA Statistics (64-bit version).

Description:

This structure contains 64-bit version of the statistics on the Cryptographic DSA operations. Statistics are set to zero when the component is initialized, and are collected per instance.

Definition at line 451 of file cpa_cy_dsa.h.

5.13.2 Field Documentation

5.13.2.1 numDsaPParamGenRequests

`Cpa64U _CpaCyDsaStats64::numDsaPParamGenRequests`

Total number of successful DSA P parameter generation requests.

Definition at line 452 of file cpa_cy_dsa.h.

5.13.2.2 numDsaPParamGenRequestErrors

`Cpa64U _CpaCyDsaStats64::numDsaPParamGenRequestErrors`

Total number of DSA P parameter generation requests that had an error and could not be processed.

Definition at line 454 of file cpa_cy_dsa.h.

5.13.2.3 numDsaPParamGenCompleted

`Cpa64U _CpaCyDsaStats64::numDsaPParamGenCompleted`

Total number of DSA P parameter generation operations that completed successfully.

Definition at line 457 of file cpa_cy_dsa.h.

5.13.2.4 numDsaPParamGenCompletedErrors

`Cpa64U _CpaCyDsaStats64::numDsaPParamGenCompletedErrors`

Total number of DSA P parameter generation operations that could not be completed successfully due to errors.

Definition at line 460 of file cpa_cy_dsa.h.

5.13.2.5 numDsaGParamGenRequests

`Cpa64U _CpaCyDsaStats64::numDsaGParamGenRequests`

Total number of successful DSA G parameter generation requests.

Definition at line 463 of file cpa_cy_dsa.h.

5.13.2.6 numDsaGParamGenRequestErrors

`Cpa64U _CpaCyDsaStats64::numDsaGParamGenRequestErrors`

Total number of DSA G parameter generation requests that had an error and could not be processed.

Definition at line 465 of file cpa_cy_dsa.h.

5.13.2.7 numDsaGParamGenCompleted

`Cpa64U _CpaCyDsaStats64::numDsaGParamGenCompleted`

Total number of DSA G parameter generation operations that completed successfully.

Definition at line 468 of file cpa_cy_dsa.h.

5.13.2.8 numDsaGParamGenCompletedErrors

`Cpa64U _CpaCyDsaStats64::numDsaGParamGenCompletedErrors`

Total number of DSA G parameter generation operations that could not be completed successfully due to errors.

Definition at line 471 of file cpa_cy_dsa.h.

5.13.2.9 numDsaYParamGenRequests

`Cpa64U _CpaCyDsaStats64::numDsaYParamGenRequests`

Total number of successful DSA Y parameter generation requests.

Definition at line 474 of file cpa_cy_dsa.h.

5.13.2.10 numDsaYParamGenRequestErrors

`Cpa64U _CpaCyDsaStats64::numDsaYParamGenRequestErrors`

Total number of DSA Y parameter generation requests that had an error and could not be processed.

Definition at line 476 of file cpa_cy_dsa.h.

5.13.2.11 numDsaYParamGenCompleted

`Cpa64U _CpaCyDsaStats64::numDsaYParamGenCompleted`

Total number of DSA Y parameter generation operations that completed successfully.

Definition at line 479 of file cpa_cy_dsa.h.

5.13.2.12 numDsaYParamGenCompletedErrors

`Cpa64U _CpaCyDsaStats64::numDsaYParamGenCompletedErrors`

Total number of DSA Y parameter generation operations that could not be completed successfully due to errors.

Definition at line 482 of file cpa_cy_dsa.h.

5.13.2.13 numDsaRSignRequests

`Cpa64U _CpaCyDsaStats64::numDsaRSignRequests`

Total number of successful DSA R sign generation requests.

Definition at line 485 of file cpa_cy_dsa.h.

5.13.2.14 numDsaRSignRequestErrors

Cpa64U _CpaCyDsaStats64::numDsaRSignRequestErrors

Total number of DSA R sign requests that had an error and could not be processed.

Definition at line 487 of file cpa_cy_dsa.h.

5.13.2.15 numDsaRSignCompleted

Cpa64U _CpaCyDsaStats64::numDsaRSignCompleted

Total number of DSA R sign operations that completed successfully.

Definition at line 490 of file cpa_cy_dsa.h.

5.13.2.16 numDsaRSignCompletedErrors

Cpa64U _CpaCyDsaStats64::numDsaRSignCompletedErrors

Total number of DSA R sign operations that could not be completed successfully due to errors.

Definition at line 493 of file cpa_cy_dsa.h.

5.13.2.17 numDsaSSignRequests

Cpa64U _CpaCyDsaStats64::numDsaSSignRequests

Total number of successful DSA S sign generation requests.

Definition at line 496 of file cpa_cy_dsa.h.

5.13.2.18 numDsaSSignRequestErrors

Cpa64U _CpaCyDsaStats64::numDsaSSignRequestErrors

Total number of DSA S sign requests that had an error and could not be processed.

Definition at line 498 of file cpa_cy_dsa.h.

5.13.2.19 numDsaSSignCompleted

`Cpa64U _CpaCyDsaStats64::numDsaSSignCompleted`

Total number of DSA S sign operations that completed successfully.

Definition at line 501 of file cpa_cy_dsa.h.

5.13.2.20 numDsaSSignCompletedErrors

`Cpa64U _CpaCyDsaStats64::numDsaSSignCompletedErrors`

Total number of DSA S sign operations that could not be completed successfully due to errors.

Definition at line 504 of file cpa_cy_dsa.h.

5.13.2.21 numDsaRSSignRequests

`Cpa64U _CpaCyDsaStats64::numDsaRSSignRequests`

Total number of successful DSA RS sign generation requests.

Definition at line 507 of file cpa_cy_dsa.h.

5.13.2.22 numDsaRSSignRequestErrors

`Cpa64U _CpaCyDsaStats64::numDsaRSSignRequestErrors`

Total number of DSA RS sign requests that had an error and could not be processed.

Definition at line 509 of file cpa_cy_dsa.h.

5.13.2.23 numDsaRSSignCompleted

`Cpa64U _CpaCyDsaStats64::numDsaRSSignCompleted`

Total number of DSA RS sign operations that completed successfully.

Definition at line 512 of file cpa_cy_dsa.h.

5.13.2.24 numDsaRSSignCompletedErrors

`Cpa64U _CpaCyDsaStats64::numDsaRSSignCompletedErrors`

Total number of DSA RS sign operations that could not be completed successfully due to errors.

Definition at line 515 of file cpa_cy_dsa.h.

5.13.2.25 numDsaVerifyRequests

`Cpa64U _CpaCyDsaStats64::numDsaVerifyRequests`

Total number of successful DSA verify generation requests.

Definition at line 518 of file cpa_cy_dsa.h.

5.13.2.26 numDsaVerifyRequestErrors

`Cpa64U _CpaCyDsaStats64::numDsaVerifyRequestErrors`

Total number of DSA verify requests that had an error and could not be processed.

Definition at line 520 of file cpa_cy_dsa.h.

5.13.2.27 numDsaVerifyCompleted

`Cpa64U _CpaCyDsaStats64::numDsaVerifyCompleted`

Total number of DSA verify operations that completed successfully.

Definition at line 523 of file cpa_cy_dsa.h.

5.13.2.28 numDsaVerifyCompletedErrors

`Cpa64U _CpaCyDsaStats64::numDsaVerifyCompletedErrors`

Total number of DSA verify operations that could not be completed successfully due to errors.

Definition at line 526 of file cpa_cy_dsa.h.

5.13.2.29 numDsaVerifyFailures

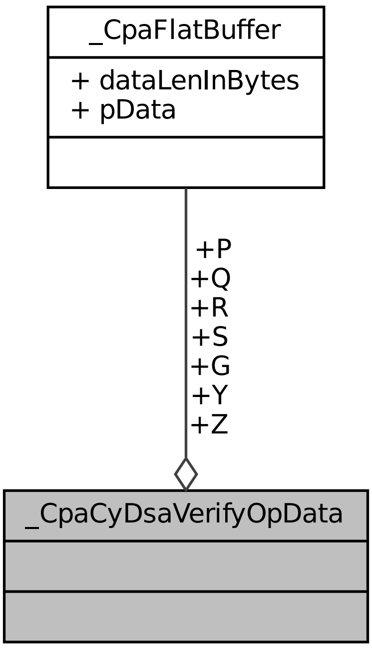
```
Cpa64U _CpaCyDsaStats64::numDsaVerifyFailures
```

Total number of DSA verify operations that executed successfully but the outcome of the test was that the verification failed. Note that this does not indicate an error.

Definition at line 529 of file cpa_cy_dsa.h.

5.14 _CpaCyDsaVerifyOpData Struct Reference

Collaboration diagram for _CpaCyDsaVerifyOpData:



Data Fields

- CpaFlatBuffer P
- CpaFlatBuffer Q
- CpaFlatBuffer G
- CpaFlatBuffer Y
- CpaFlatBuffer Z
- CpaFlatBuffer R
- CpaFlatBuffer S

5.14.1 Detailed Description

DSA Verify Operation Data.

Description:

This structure contains the operation data for the cpaCyDsaVerify function. The client MUST allocate the memory for this structure and the items pointed to by this structure. When the structure is passed into the function, ownership of the memory passes to the function. Ownership of the memory returns to the client when this structure is returned in the callback function.

For optimal performance all data SHOULD be 8-byte aligned.

All values in this structure are required to be in Most Significant Byte first order, e.g. P.pData[0] = MSB.

Note

If the client modifies or frees the memory referenced in this structure after it has been submitted to the cpaCyDsaVerify function, and before it has been returned in the callback, undefined behavior will result.

See also

cpaCyDsaVerify()

Definition at line 324 of file cpa_cy_dsa.h.

5.14.2 Field Documentation

5.14.2.1 P

`CpaFlatBuffer _CpaCyDsaVerifyOpData::P`

DSA group parameter p

Definition at line 325 of file cpa_cy_dsa.h.

5.14.2.2 Q

`CpaFlatBuffer _CpaCyDsaVerifyOpData::Q`

DSA group parameter q

Definition at line 327 of file cpa_cy_dsa.h.

5.14.2.3 G

`CpaFlatBuffer _CpaCyDsaVerifyOpData::G`

DSA group parameter g

Definition at line 329 of file cpa_cy_dsa.h.

5.14.2.4 Y

`CpaFlatBuffer _CpaCyDsaVerifyOpData::Y`

DSA public key y

Definition at line 331 of file cpa_cy_dsa.h.

5.14.2.5 Z

`CpaFlatBuffer _CpaCyDsaVerifyOpData::Z`

The leftmost min(N, outlen) bits of Hash(M'), where:

- N is the bit length of q
- outlen is the bit length of the hash function output block
- M is the message to be signed

Definition at line 333 of file cpa_cy_dsa.h.

5.14.2.6 R

`CpaFlatBuffer _CpaCyDsaVerifyOpData::R`

DSA message signature r

Definition at line 339 of file cpa_cy_dsa.h.

5.14.2.7 S

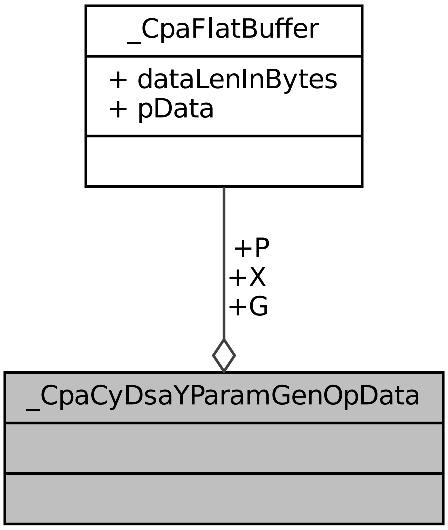
`CpaFlatBuffer _CpaCyDsaVerifyOpData::S`

DSA message signature s

Definition at line 341 of file cpa_cy_dsa.h.

5.15 _CpaCyDsaYParamGenOpData Struct Reference

Collaboration diagram for _CpaCyDsaYParamGenOpData:



Data Fields

- CpaFlatBuffer P
- CpaFlatBuffer G
- CpaFlatBuffer X

5.15.1 Detailed Description

DSA Y Parameter Generation Operation Data.

Description:

This structure contains the operation data for the cpaCyDsaGenYParam function. The client MUST allocate the memory for this structure and the items pointed to by this structure. When the structure is passed into the function, ownership of the memory passes to the function. Ownership of the memory returns to the client when this structure is returned in the callback function.

For optimal performance all data SHOULD be 8-byte aligned.

All values in this structure are required to be in Most Significant Byte first order, e.g. P.pData[0] = MSB.

Note

If the client modifies or frees the memory referenced in this structure after it has been submitted to the cpaCyDsaGenYParam function, and before it has been returned in the callback, undefined behavior will result.

See also

cpaCyDsaGenYParam()

Definition at line 160 of file cpa_cy_dsa.h.

5.15.2 Field Documentation

5.15.2.1 P

`CpaFlatBuffer _CpaCyDsaYParamGenOpData::P`

DSA group parameter p

Definition at line 161 of file cpa_cy_dsa.h.

5.15.2.2 G

`CpaFlatBuffer _CpaCyDsaYParamGenOpData::G`

DSA group parameter g

Definition at line 163 of file cpa_cy_dsa.h.

5.15.2.3 X

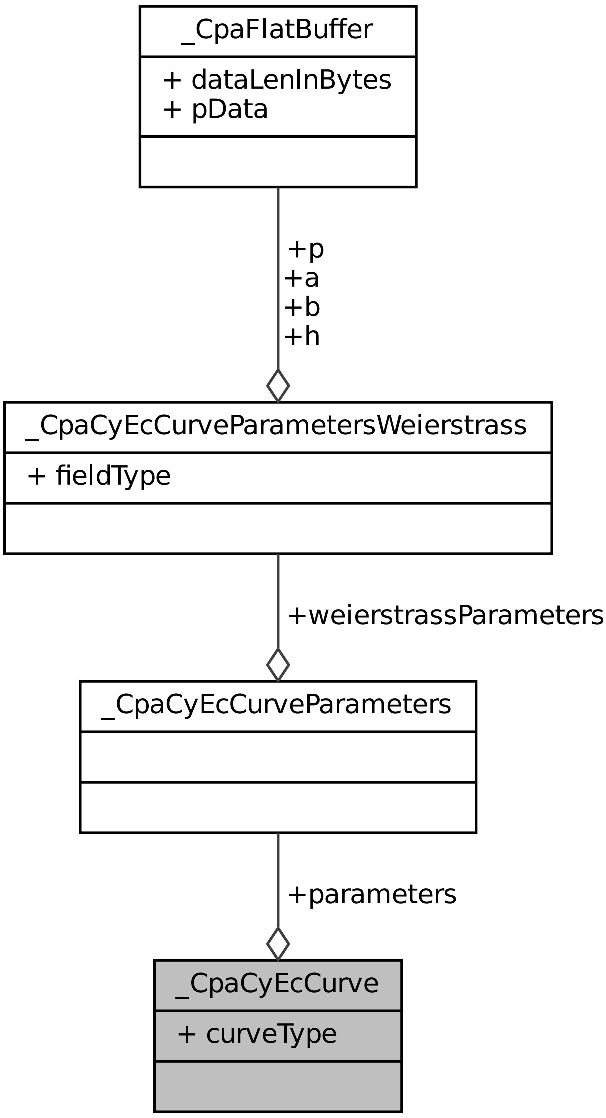
`CpaFlatBuffer _CpaCyDsaYParamGenOpData::X`

DSA private key x

Definition at line 165 of file cpa_cy_dsa.h.

5.16 _CpaCyEcCurve Struct Reference

Collaboration diagram for _CpaCyEcCurve:



Data Fields

- CpaCyEcCurveType **curveType**
- CpaCyEcCurveParameters **parameters**

5.16.1 Detailed Description

Unified curve parameters.

Description:

This structure provides a single data type that can describe a number of different curve types. The intention is to add further curve types in the future, thus the union field will allow for that expansion.

The client MUST allocate the memory for this structure and the items pointed to by this structure. When the structure is passed into the function, ownership of the memory passes to the function. Ownership of the memory returns to the client when this structure is returned in the callback function.

For optimal performance all data buffers SHOULD be 8-byte aligned.

Note

If the client modifies or frees the memory referenced in this structure after it has been submitted to the function, and before it has been returned in the callback, undefined behavior will result.

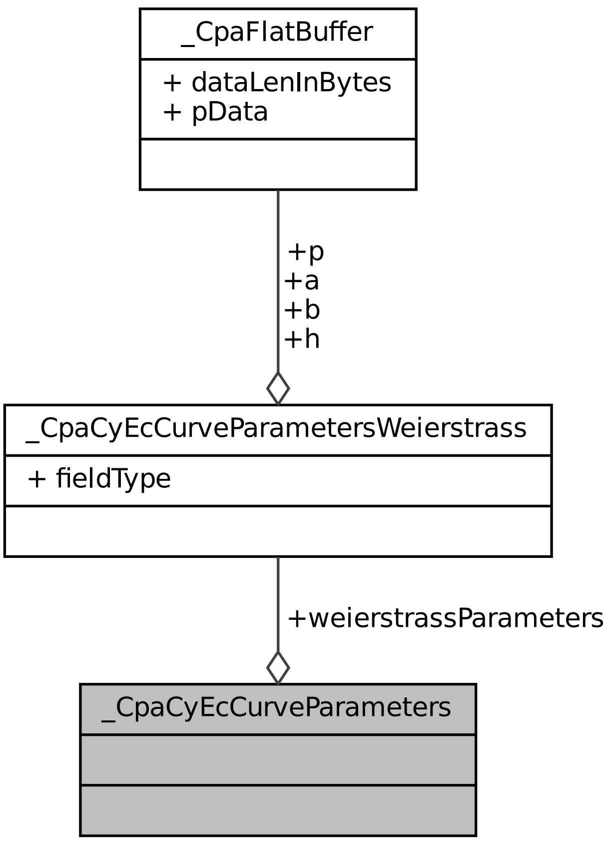
See also

CpaCyEcCurveParameters cpaCyEcGenericPointMultiply cpaCyEcGenericPointVerify

Definition at line 283 of file cpa_cy_ec.h.

5.17 _CpaCyEcCurveParameters Union Reference

Collaboration diagram for _CpaCyEcCurveParameters:



Data Fields

- CpaCyEcCurveParametersWeierstrass **weierstrassParameters**

5.17.1 Detailed Description

Union characterised by a specific curve.

Description:

This union allows for the characterisation of different curve types encapsulted in one data type. The intention is that new curve types will be added in the future.

Note

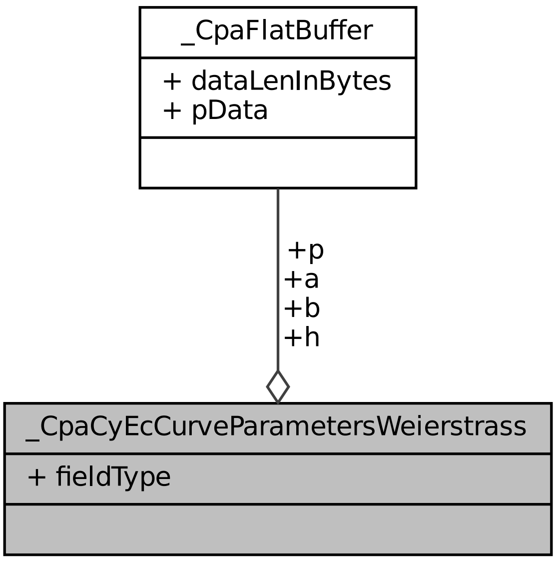
See also

CpaCyEcCurveParametersWeierstrass

Definition at line 248 of file cpa_cy_ec.h.

5.18 _CpaCyEcCurveParametersWeierstrass Struct Reference

Collaboration diagram for _CpaCyEcCurveParametersWeierstrass:



Data Fields

- CpaCyEcFieldType fieldType
- CpaFlatBuffer p
- CpaFlatBuffer a
- CpaFlatBuffer b
- CpaFlatBuffer h

5.18.1 Detailed Description

Curve parameters for a Weierstrass type curve.

Description:

This structure contains curve parameters for Weierstrass type curve: y^2 = x^3 + ax + b The client MUST allocate the memory for this structure When the structure is passed into the function, ownership of the memory passes to the function. Ownership of the memory returns to the client when this structure is returned in the callback function.

For optimal performance all data buffers SHOULD be 8-byte aligned. The legend used in this structure is borrowed from RFC7748

Note

If the client modifies or frees the memory referenced in this structure after it has been submitted to the function, and before it has been returned in the callback, undefined behavior will result.

See also

CpaCyEcCurveParameters CpaCyEcFieldType

Definition at line 218 of file cpa_cy_ec.h.

5.18.2 Field Documentation

5.18.2.1 fieldType

```
CpaCyEcFieldType _CpaCyEcCurveParametersWeierstrass::fieldType
```

Prime or Binary

Definition at line 220 of file cpa_cy_ec.h.

5.18.2.2 p

```
CpaFlatBuffer _CpaCyEcCurveParametersWeierstrass::p
```

Prime modulus or irreducible polynomial over GF(2^m)

Definition at line 222 of file cpa_cy_ec.h.

5.18.2.3 a

```
CpaFlatBuffer _CpaCyEcCurveParametersWeierstrass::a
```

a coefficient

Definition at line 224 of file cpa_cy_ec.h.

5.18.2.4 b

```
CpaFlatBuffer _CpaCyEcCurveParametersWeierstrass::b
```

b coefficient

Definition at line 226 of file cpa_cy_ec.h.

5.18.2.5 h

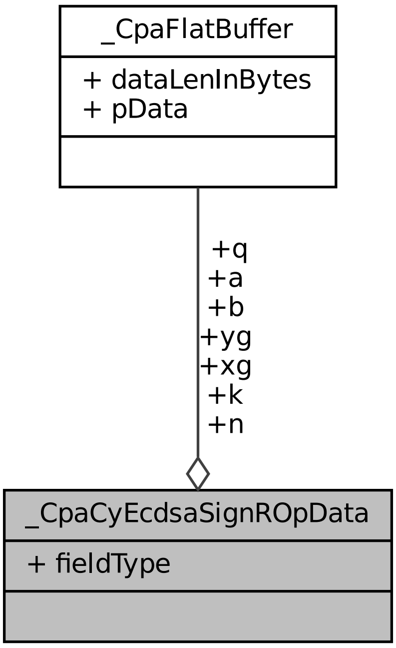
```
CpaFlatBuffer _CpaCyEcCurveParametersWeierstrass::h
```

Cofactor

Definition at line 228 of file cpa_cy_ec.h.

5.19 _CpaCyEcdsaSignROpData Struct Reference

Collaboration diagram for _CpaCyEcdsaSignROpData:



Data Fields

- CpaFlatBuffer xg
- CpaFlatBuffer yg
- CpaFlatBuffer n
- CpaFlatBuffer q
- CpaFlatBuffer a
- CpaFlatBuffer b
- CpaFlatBuffer k
- CpaCyEcFieldType fieldType

5.19.1 Detailed Description

ECDSA Sign R Operation Data.

Description:

This structure contains the operation data for the cpaCyEcdsaSignR function. The client MUST allocate the memory for this structure and the items pointed to by this structure. When the structure is passed into the function, ownership of the memory passes to the function. Ownership of the memory returns to the client when this structure is returned in the callback function.

For optimal performance all data buffers SHOULD be 8-byte aligned.

All values in this structure are required to be in Most Significant Byte first order, e.g. a.pData[0] = MSB.

Note

If the client modifies or frees the memory referenced in this structure after it has been submitted to the cpaCyEcdsaSignR function, and before it has been returned in the callback, undefined behavior will result.

See also

cpaCyEcdsaSignR()

Definition at line 70 of file cpa_cy_ecdsa.h.

5.19.2 Field Documentation

5.19.2.1 xg

`CpaFlatBuffer _CpaCyEcdsaSignROpData::xg`

x coordinate of base point G

Definition at line 71 of file cpa_cy_ecdsa.h.

5.19.2.2 yg

`CpaFlatBuffer _CpaCyEcdsaSignROpData::yg`

y coordinate of base point G

Definition at line 73 of file cpa_cy_ecdsa.h.

5.19.2.3 n

`CpaFlatBuffer _CpaCyEcdsaSignROpData::n`

order of the base point G, which shall be prime

Definition at line 75 of file cpa_cy_ecdsa.h.

5.19.2.4 q

`CpaFlatBuffer _CpaCyEcdsaSignROpData::q`

prime modulus or irreducible polynomial over GF(2^r)

Definition at line 77 of file cpa_cy_ecdsa.h.

5.19.2.5 a

`CpaFlatBuffer _CpaCyEcdsaSignROpData::a`

a elliptic curve coefficient

Definition at line 79 of file cpa_cy_ecdsa.h.

5.19.2.6 b

`CpaFlatBuffer _CpaCyEcdsaSignROpData::b`

b elliptic curve coefficient

Definition at line 81 of file cpa_cy_ecdsa.h.

5.19.2.7 k

`CpaFlatBuffer _CpaCyEcdsaSignROpData::k`

random value ($k > 0$ and $k < n$)

Definition at line 83 of file cpa_cy_ecdsa.h.

5.19.2.8 fieldType

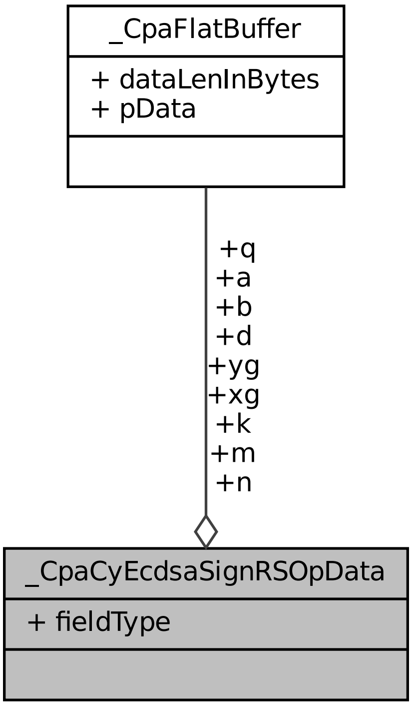
`CpaCyEcFieldType _CpaCyEcdsaSignROpData::fieldType`

field type for the operation

Definition at line 86 of file cpa_cy_ecdsa.h.

5.20 _CpaCyEcdsaSignRSOpData Struct Reference

Collaboration diagram for _CpaCyEcdsaSignRSOpData:



Data Fields

- CpaFlatBuffer xg
- CpaFlatBuffer yg
- CpaFlatBuffer n
- CpaFlatBuffer q
- CpaFlatBuffer a
- CpaFlatBuffer b
- CpaFlatBuffer k
- CpaFlatBuffer m
- CpaFlatBuffer d
- CpaCyEcFieldType fieldType

5.20.1 Detailed Description

ECDSA Sign R & S Operation Data.

Description:

This structure contains the operation data for the cpaCyEcdsaSignRS function. The client MUST allocate the memory for this structure and the items pointed to by this structure. When the structure is passed into the function, ownership of the memory passes to the function. Ownership of the memory returns to the client when this structure is returned in the callback function.

For optimal performance all data buffers SHOULD be 8-byte aligned.

All values in this structure are required to be in Most Significant Byte first order, e.g. a.pData[0] = MSB.

Note

If the client modifies or frees the memory referenced in this structure after it has been submitted to the cpaCyEcdsaSignRS function, and before it has been returned in the callback, undefined behavior will result.

See also

cpaCyEcdsaSignRS()

Definition at line 161 of file cpa_cy_ecdsa.h.

5.20.2 Field Documentation

5.20.2.1 xg

`CpaFlatBuffer _CpaCyEcdsaSignRSOpData::xg`

x coordinate of base point G

Definition at line 162 of file cpa_cy_ecdsa.h.

5.20.2.2 yg

`CpaFlatBuffer _CpaCyEcdsaSignRSOpData::yg`

y coordinate of base point G

Definition at line 164 of file cpa_cy_ecdsa.h.

5.20.2.3 n

`CpaFlatBuffer _CpaCyEcdsaSignRSOpData::n`

order of the base point G, which shall be prime

Definition at line 166 of file cpa_cy_ecdsa.h.

5.20.2.4 q

`CpaFlatBuffer _CpaCyEcdsaSignRSOpData::q`

prime modulus or irreducible polynomial over GF(2^r)

Definition at line 168 of file cpa_cy_ecdsa.h.

5.20.2.5 a

`CpaFlatBuffer _CpaCyEcdsaSignRSOpData::a`

a elliptic curve coefficient

Definition at line 170 of file cpa_cy_ecdsa.h.

5.20.2.6 b

`CpaFlatBuffer _CpaCyEcdsaSignRSOpData::b`

b elliptic curve coefficient

Definition at line 172 of file cpa_cy_ecdsa.h.

5.20.2.7 k

`CpaFlatBuffer _CpaCyEcdsaSignRSOpData::k`

random value ($k > 0$ and $k < n$)

Definition at line 174 of file cpa_cy_ecdsa.h.

5.20.2.8 m

CpaFlatBuffer _CpaCyEcdsaSignRSOpData::m

digest of the message to be signed

Definition at line 176 of file cpa_cy_ecdsa.h.

5.20.2.9 d

CpaFlatBuffer _CpaCyEcdsaSignRSOpData::d

private key

Definition at line 178 of file cpa_cy_ecdsa.h.

5.20.2.10 fieldType

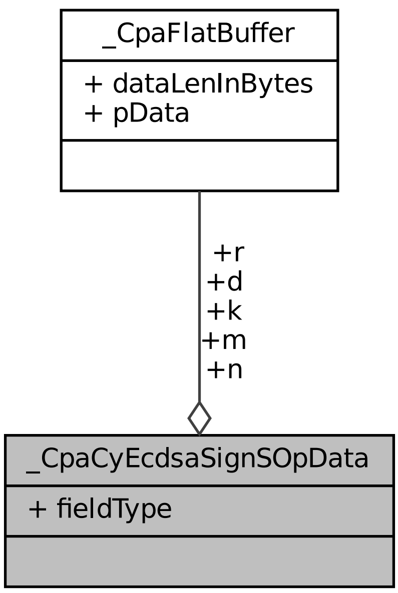
CpaCyEcFieldType _CpaCyEcdsaSignRSOpData::fieldType

field type for the operation

Definition at line 180 of file cpa_cy_ecdsa.h.

5.21 _CpaCyEcdsaSignSOpData Struct Reference

Collaboration diagram for _CpaCyEcdsaSignSOpData:



Data Fields

- CpaFlatBuffer m
- CpaFlatBuffer d
- CpaFlatBuffer r
- CpaFlatBuffer k
- CpaFlatBuffer n
- CpaCyEcFieldType fieldType

5.21.1 Detailed Description

ECDSA Sign S Operation Data.

Description:

This structure contains the operation data for the cpaCyEcdsaSignS function. The client MUST allocate the memory for this structure and the items pointed to by this structure. When the structure is passed into the function, ownership of the memory passes to the function. Ownership of the memory returns to the client when this structure is returned in the callback function.

For optimal performance all data buffers SHOULD be 8-byte aligned.

All values in this structure are required to be in Most Significant Byte first order, e.g. a.pData[0] = MSB.

Note

If the client modifies or frees the memory referenced in this structure after it has been submitted to the cpaCyEcdsaSignS function, and before it has been returned in the callback, undefined behavior will result.

See also

cpaCyEcdsaSignS()

Definition at line 118 of file cpa_cy_ecdsa.h.

5.21.2 Field Documentation

5.21.2.1 m

`CpaFlatBuffer _CpaCyEcdsaSignSOpData::m`

digest of the message to be signed

Definition at line 119 of file cpa_cy_ecdsa.h.

5.21.2.2 d

`CpaFlatBuffer _CpaCyEcdsaSignSOpData::d`

private key

Definition at line 121 of file cpa_cy_ecdsa.h.

5.21.2.3 r

`CpaFlatBuffer _CpaCyEcdsaSignSOpData::r`

Ecdsa r signature value

Definition at line 123 of file cpa_cy_ecdsa.h.

5.21.2.4 k

`CpaFlatBuffer _CpaCyEcdsaSignSOpData::k`

random value ($k > 0$ and $k < n$)

Definition at line 125 of file cpa_cy_ecdsa.h.

5.21.2.5 n

`CpaFlatBuffer _CpaCyEcdsaSignSOpData::n`

order of the base point G, which shall be prime

Definition at line 127 of file cpa_cy_ecdsa.h.

5.21.2.6 fieldType

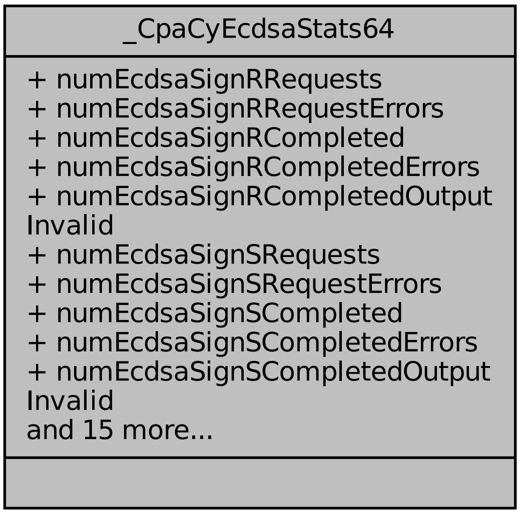
`CpaCyEcFieldType _CpaCyEcdsaSignSOpData::fieldType`

field type for the operation

Definition at line 129 of file cpa_cy_ecdsa.h.

5.22 _CpaCyEcdsaStats64 Struct Reference

Collaboration diagram for _CpaCyEcdsaStats64:



Data Fields

- Cpa64U numEcdsaSignRRequests
- Cpa64U numEcdsaSignRRequestErrors
- Cpa64U numEcdsaSignRCompleted
- Cpa64U numEcdsaSignRCompletedErrors
- Cpa64U numEcdsaSignRCompletedOutputInvalid
- Cpa64U numEcdsaSignSRequests
- Cpa64U numEcdsaSignSRequestErrors
- Cpa64U numEcdsaSignSCompleted
- Cpa64U numEcdsaSignSCompletedErrors
- Cpa64U numEcdsaSignSCompletedOutputInvalid
- Cpa64U numEcdsaSignRSRequests
- Cpa64U numEcdsaSignRSRequestErrors
- Cpa64U numEcdsaSignRSCompleted
- Cpa64U numEcdsaSignRSCompletedErrors
- Cpa64U numEcdsaSignRSCompletedOutputInvalid
- Cpa64U numEcdsaVerifyRequests
- Cpa64U numEcdsaVerifyRequestErrors
- Cpa64U numEcdsaVerifyCompleted
- Cpa64U numEcdsaVerifyCompletedErrors
- Cpa64U numEcdsaVerifyCompletedOutputInvalid
- Cpa64U numKptEcdsaSignRSCompletedOutputInvalid
- Cpa64U numKptEcdsaSignRSCompleted
- Cpa64U numKptEcdsaSignRSRequests
- Cpa64U numKptEcdsaSignRSRequestErrors
- Cpa64U numKptEcdsaSignRSCompletedErrors

5.22.1 Detailed Description

Cryptographic ECDSA Statistics.

Description:

This structure contains statistics on the Cryptographic ECDSA operations. Statistics are set to zero when the component is initialized, and are collected per instance.

Definition at line 250 of file cpa_cy_ecdsa.h.

5.22.2 Field Documentation

5.22.2.1 numEcdsaSignRRequests

`Cpa64U _CpaCyEcdsaStats64::numEcdsaSignRRequests`

Total number of ECDSA Sign R operation requests.

Definition at line 251 of file cpa_cy_ecdsa.h.

5.22.2.2 numEcdsaSignRRequestErrors

`Cpa64U _CpaCyEcdsaStats64::numEcdsaSignRRequestErrors`

Total number of ECDSA Sign R operation requests that had an error and could not be processed.

Definition at line 253 of file cpa_cy_ecdsa.h.

5.22.2.3 numEcdsaSignRCompleted

`Cpa64U _CpaCyEcdsaStats64::numEcdsaSignRCompleted`

Total number of ECDSA Sign R operation requests that completed successfully.

Definition at line 256 of file cpa_cy_ecdsa.h.

5.22.2.4 numEcdsaSignRCompletedErrors

`Cpa64U _CpaCyEcdsaStats64::numEcdsaSignRCompletedErrors`

Total number of ECDSA Sign R operation requests that could not be completed successfully due to errors.

Definition at line 259 of file cpa_cy_ecdsa.h.

5.22.2.5 numEcdsaSignRCompletedOutputInvalid

`Cpa64U _CpaCyEcdsaStats64::numEcdsaSignRCompletedOutputInvalid`

Total number of ECDSA Sign R operation requests could not be completed successfully due to an invalid output. Note that this does not indicate an error.

Definition at line 262 of file cpa_cy_ecdsa.h.

5.22.2.6 numEcdsaSignSRequests

`Cpa64U _CpaCyEcdsaStats64::numEcdsaSignSRequests`

Total number of ECDSA Sign S operation requests.

Definition at line 266 of file cpa_cy_ecdsa.h.

5.22.2.7 numEcdsaSignSRequestErrors

`Cpa64U _CpaCyEcdsaStats64::numEcdsaSignSRequestErrors`

Total number of ECDSA Sign S operation requests that had an error and could not be processed.

Definition at line 268 of file cpa_cy_ecdsa.h.

5.22.2.8 numEcdsaSignSCompleted

`Cpa64U _CpaCyEcdsaStats64::numEcdsaSignSCompleted`

Total number of ECDSA Sign S operation requests that completed successfully.

Definition at line 271 of file cpa_cy_ecdsa.h.

5.22.2.9 numEcdsaSignSCompletedErrors

`Cpa64U _CpaCyEcdsaStats64::numEcdsaSignSCompletedErrors`

Total number of ECDSA Sign S operation requests that could not be completed successfully due to errors.

Definition at line 274 of file cpa_cy_ecdsa.h.

5.22.2.10 numEcdsaSignSCompletedOutputInvalid

`Cpa64U _CpaCyEcdsaStats64::numEcdsaSignSCompletedOutputInvalid`

Total number of ECDSA Sign S operation requests could not be completed successfully due to an invalid output. Note that this does not indicate an error.

Definition at line 277 of file cpa_cy_ecdsa.h.

5.22.2.11 numEcdsaSignRSRequests

`Cpa64U _CpaCyEcdsaStats64::numEcdsaSignRSRequests`

Total number of ECDSA Sign R & S operation requests.

Definition at line 281 of file cpa_cy_ecdsa.h.

5.22.2.12 numEcdsaSignRSRequestErrors

`Cpa64U _CpaCyEcdsaStats64::numEcdsaSignRSRequestErrors`

Total number of ECDSA Sign R & S operation requests that had an error and could not be processed.

Definition at line 283 of file cpa_cy_ecdsa.h.

5.22.2.13 numEcdsaSignRSCompleted

`Cpa64U _CpaCyEcdsaStats64::numEcdsaSignRSCompleted`

Total number of ECDSA Sign R & S operation requests that completed successfully.

Definition at line 286 of file cpa_cy_ecdsa.h.

5.22.2.14 numEcdsaSignRSCompletedErrors

`Cpa64U _CpaCyEcdsaStats64::numEcdsaSignRSCompletedErrors`

Total number of ECDSA Sign R & S operation requests that could not be completed successfully due to errors.

Definition at line 289 of file cpa_cy_ecdsa.h.

5.22.2.15 numEcdsaSignRSCompletedOutputInvalid

`Cpa64U _CpaCyEcdsaStats64::numEcdsaSignRSCompletedOutputInvalid`

Total number of ECDSA Sign R & S operation requests could not be completed successfully due to an invalid output. Note that this does not indicate an error.

Definition at line 292 of file cpa_cy_ecdsa.h.

5.22.2.16 numEcdsaVerifyRequests

`Cpa64U _CpaCyEcdsaStats64::numEcdsaVerifyRequests`

Total number of ECDSA Verification operation requests.

Definition at line 296 of file cpa_cy_ecdsa.h.

5.22.2.17 numEcdsaVerifyRequestErrors

`Cpa64U _CpaCyEcdsaStats64::numEcdsaVerifyRequestErrors`

Total number of ECDSA Verification operation requests that had an error and could not be processed.

Definition at line 298 of file cpa_cy_ecdsa.h.

5.22.2.18 numEcdsaVerifyCompleted

`Cpa64U _CpaCyEcdsaStats64::numEcdsaVerifyCompleted`

Total number of ECDSA Verification operation requests that completed successfully.

Definition at line 301 of file cpa_cy_ecdsa.h.

5.22.2.19 numEcdsaVerifyCompletedErrors

`Cpa64U _CpaCyEcdsaStats64::numEcdsaVerifyCompletedErrors`

Total number of ECDSA Verification operation requests that could not be completed successfully due to errors.

Definition at line 304 of file cpa_cy_ecdsa.h.

5.22.2.20 numEcdsaVerifyCompletedOutputInvalid

`Cpa64U _CpaCyEcdsaStats64::numEcdsaVerifyCompletedOutputInvalid`

Total number of ECDSA Verification operation requests that resulted in an invalid output. Note that this does not indicate an error.

Definition at line 307 of file cpa_cy_ecdsa.h.

5.22.2.21 numKptEcdsaSignRSCompletedOutputInvalid

`Cpa64U _CpaCyEcdsaStats64::numKptEcdsaSignRSCompletedOutputInvalid`

Total number of KPT ECDSA Sign R & S operation requests could not be completed successfully due to an invalid output. Note that this does not indicate an error.

Definition at line 311 of file cpa_cy_ecdsa.h.

5.22.2.22 numKptEcdsaSignRSCompleted

`Cpa64U _CpaCyEcdsaStats64::numKptEcdsaSignRSCompleted`

Total number of KPT ECDSA Sign R & S operation requests that completed successfully.

Definition at line 315 of file cpa_cy_ecdsa.h.

5.22.2.23 numKptEcdsaSignRSRequests

`Cpa64U _CpaCyEcdsaStats64::numKptEcdsaSignRSRequests`

Total number of KPT ECDSA Sign R & S operation requests.

Definition at line 318 of file cpa_cy_ecdsa.h.

5.22.2.24 numKptEcdsaSignRSRequestErrors

`Cpa64U _CpaCyEcdsaStats64::numKptEcdsaSignRSRequestErrors`

Total number of KPT ECDSA Sign R & S operation requests that had an error and could not be processed.

Definition at line 320 of file cpa_cy_ecdsa.h.

5.22.2.25 numKptEcdsaSignRSCompletedErrors

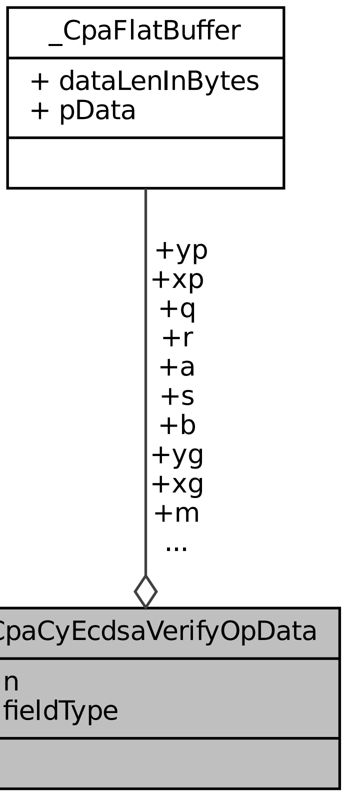
`Cpa64U _CpaCyEcdsaStats64::numKptEcdsaSignRSCompletedErrors`

Total number of KPT ECDSA Sign R & S operation requests that could not be completed successfully due to errors.

Definition at line 323 of file cpa_cy_ecdsa.h.

5.23 _CpaCyEcdsaVerifyOpData Struct Reference

Collaboration diagram for _CpaCyEcdsaVerifyOpData:



Data Fields

- CpaFlatBuffer xg
- CpaFlatBuffer yg
- CpaFlatBuffer n
- CpaFlatBuffer q
- CpaFlatBuffer a
- CpaFlatBuffer b
- CpaFlatBuffer m
- CpaFlatBuffer r
- CpaFlatBuffer s
- CpaFlatBuffer xp
- CpaFlatBuffer yp
- CpaCyEcFieldType fieldType

5.23.1 Detailed Description

ECDSA Verify Operation Data, for Public Key.

Description:

This structure contains the operation data for the CpaCyEcdsaVerify function. The client MUST allocate the memory for this structure and the items pointed to by this structure. When the structure is passed into the function, ownership of the memory passes to the function. Ownership of the memory returns to the client when this structure is returned in the callback function.

For optimal performance all data buffers SHOULD be 8-byte aligned.

All values in this structure are required to be in Most Significant Byte first order, e.g. a.pData[0] = MSB.

Note

If the client modifies or frees the memory referenced in this structure after it has been submitted to the cpaCyEcdsaVerify function, and before it has been returned in the callback, undefined behavior will result.

See also

CpaCyEcdsaVerify()

Definition at line 213 of file cpa_cy_ecdsa.h.

5.23.2 Field Documentation

5.23.2.1 xg

`CpaFlatBuffer _CpaCyEcdsaVerifyOpData::xg`

x coordinate of base point G

Definition at line 214 of file cpa_cy_ecdsa.h.

5.23.2.2 yg

`CpaFlatBuffer _CpaCyEcdsaVerifyOpData::yg`

y coordinate of base point G

Definition at line 216 of file cpa_cy_ecdsa.h.

5.23.2.3 n

`CpaFlatBuffer _CpaCyEcdsaVerifyOpData::n`

order of the base point G, which shall be prime

Definition at line 218 of file cpa_cy_ecdsa.h.

5.23.2.4 q

`CpaFlatBuffer _CpaCyEcdsaVerifyOpData::q`

prime modulus or irreducible polynomial over GF(2^r)

Definition at line 220 of file cpa_cy_ecdsa.h.

5.23.2.5 a

`CpaFlatBuffer _CpaCyEcdsaVerifyOpData::a`

a elliptic curve coefficient

Definition at line 222 of file cpa_cy_ecdsa.h.

5.23.2.6 b

`CpaFlatBuffer _CpaCyEcdsaVerifyOpData::b`

b elliptic curve coefficient

Definition at line 224 of file cpa_cy_ecdsa.h.

5.23.2.7 m

`CpaFlatBuffer _CpaCyEcdsaVerifyOpData::m`

digest of the message to be signed

Definition at line 226 of file cpa_cy_ecdsa.h.

5.23.2.8 r

`CpaFlatBuffer _CpaCyEcdsaVerifyOpData::r`

ECDSA r signature value ($r > 0$ and $r < n$)

Definition at line 228 of file cpa_cy_ecdsa.h.

5.23.2.9 s

`CpaFlatBuffer _CpaCyEcdsaVerifyOpData::s`

ECDSA s signature value ($s > 0$ and $s < n$)

Definition at line 230 of file cpa_cy_ecdsa.h.

5.23.2.10 xp

`CpaFlatBuffer _CpaCyEcdsaVerifyOpData::xp`

x coordinate of point P (public key)

Definition at line 232 of file cpa_cy_ecdsa.h.

5.23.2.11 yp

`CpaFlatBuffer _CpaCyEcdsaVerifyOpData::yp`

y coordinate of point P (public key)

Definition at line 234 of file cpa_cy_ecdsa.h.

5.23.2.12 fieldType

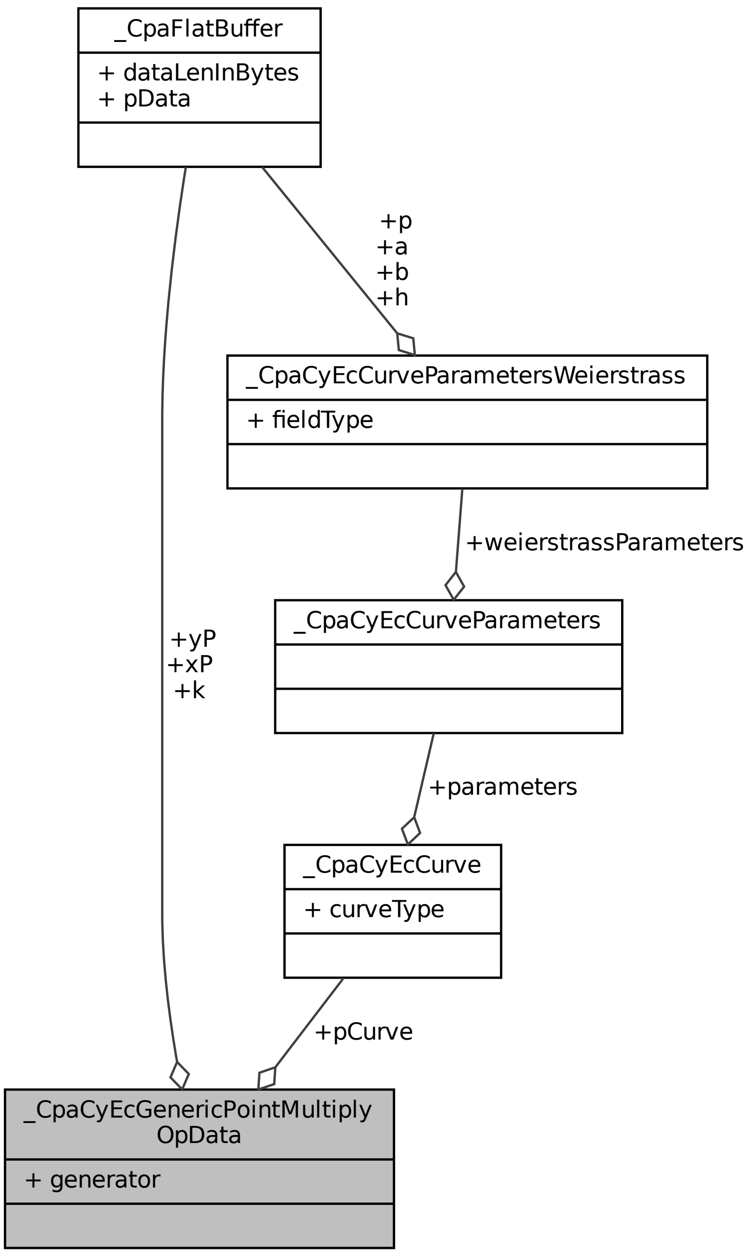
`CpaCyEcFieldType _CpaCyEcdsaVerifyOpData::fieldType`

field type for the operation

Definition at line 236 of file cpa_cy_ecdsa.h.

5.24 _CpaCyEcGenericPointMultiplyOpData Struct Reference

Collaboration diagram for _CpaCyEcGenericPointMultiplyOpData:



Data Fields

- CpaFlatBuffer **k**
- CpaFlatBuffer xP
- CpaFlatBuffer yP
- CpaCyEcCurve ∗ pCurve
- CpaBoolean generator

5.24.1 Detailed Description

Generic EC Point Multiplication Operation Data.

Description:

This structure contains a generic EC point and a multiplier for use with cpaCyEcGenericPointMultiply. This is common for representing all EC points, irrespective of curve type: Weierstrass, Montgomery and Twisted Edwards (at this time only Weierstrass are supported). The same point + multiplier format can be used when performing generator multiplication, in which case the xP, yP supplied in this structure will be ignored by QAT API library & a generator point will be inserted in their place.

For optimal performance all data buffers SHOULD be 8-byte aligned.

All values in this structure are required to be in Most Significant Byte first order, e.g. a.pData[0] = MSB.

Note

If the client modifies or frees the memory referenced in this structure after it has been submitted to the cpaCyEcGenericPointMultiply function, and before it has been returned in the callback, undefined behavior will result.

See also

cpaCyEcGenericPointMultiply()

Definition at line 368 of file cpa_cy_ec.h.

5.24.2 Field Documentation

5.24.2.1 xP

`CpaFlatBuffer _CpaCyEcGenericPointMultiplyOpData::xP`

<scalar multiplier (k > 0 and k < n)

Definition at line 371 of file cpa_cy_ec.h.

5.24.2.2 yP

`CpaFlatBuffer _CpaCyEcGenericPointMultiplyOpData::yP`

<x coordinate of public key

Definition at line 373 of file cpa_cy_ec.h.

5.24.2.3 pCurve

`CpaCyEcCurve* _CpaCyEcGenericPointMultiplyOpData::pCurve`

<y coordinate of public key

Definition at line 375 of file cpa_cy_ec.h.

5.24.2.4 generator

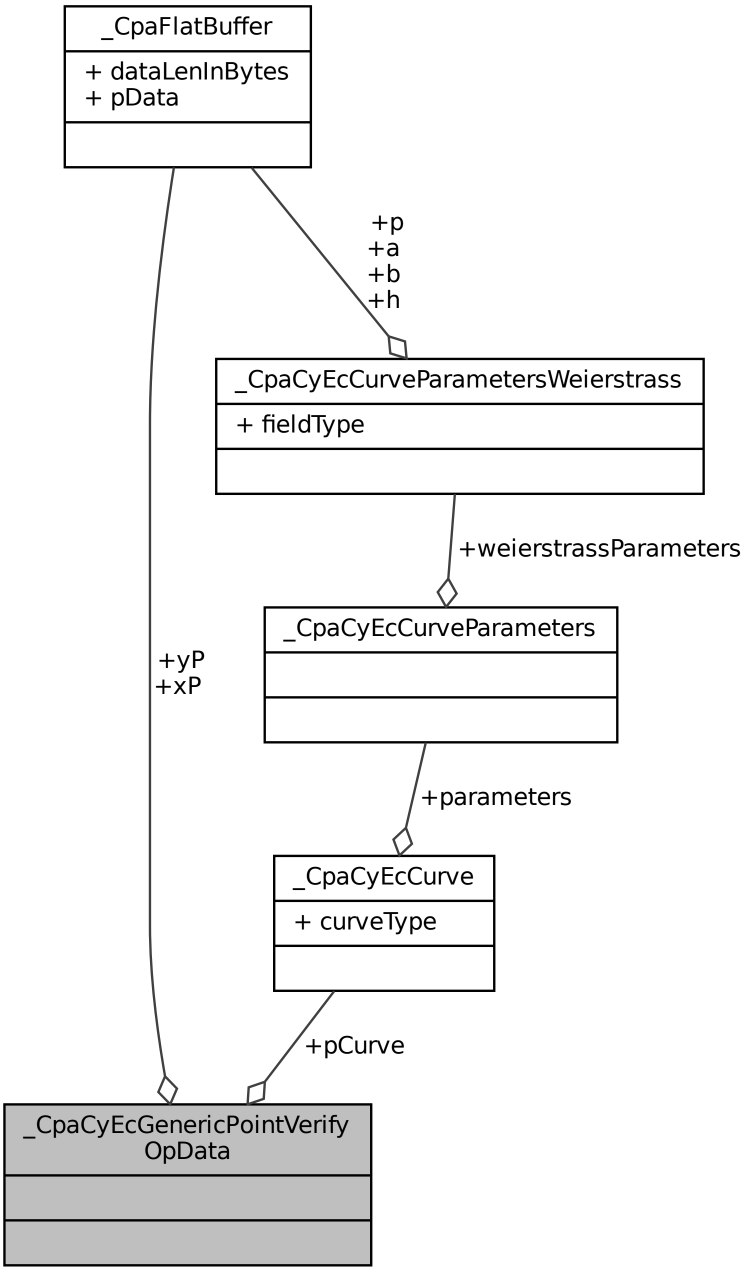
`CpaBoolean _CpaCyEcGenericPointMultiplyOpData::generator`

<curve type specific parameters

Definition at line 377 of file cpa_cy_ec.h.

5.25 _CpaCyEcGenericPointVerifyOpData Struct Reference

Collaboration diagram for _CpaCyEcGenericPointVerifyOpData:



Data Fields

- CpaFlatBuffer **xP**
- CpaFlatBuffer yP
- CpaCyEcCurve ∗ pCurve

5.25.1 Detailed Description

Generic EC Point Verify Operation Data.

Description:

This structure contains the operation data for the cpaCyEcGenericPointVerify function. This is common for representing all EC points, irrespective of curve type: Weierstrass, Montgomery and Twisted Edwards (at this time only Weierstrass are supported).

This structure contains a generic EC point, irrespective of curve type. It is used to verify when the <x,y> pair specified in the structure lies on the curve indicated in the cpaCyEcGenericPointVerify API.

For optimal performance all data buffers SHOULD be 8-byte aligned.

All values in this structure are required to be in Most Significant Byte first order, e.g. a.pData[0] = MSB.

Note

If the client modifies or frees the memory referenced in this structure after it has been submitted to the cpaCyEcGenericPointVerify function, and before it has been returned in the callback, undefined behavior will result.

See also

cpaCyEcGenericPointVerify()

Definition at line 411 of file cpa_cy_ec.h.

5.25.2 Field Documentation

5.25.2.1 yP

`CpaFlatBuffer _CpaCyEcGenericPointVerifyOpData::yP`

<x coordinate of public key

Definition at line 414 of file cpa_cy_ec.h.

5.25.2.2 pCurve

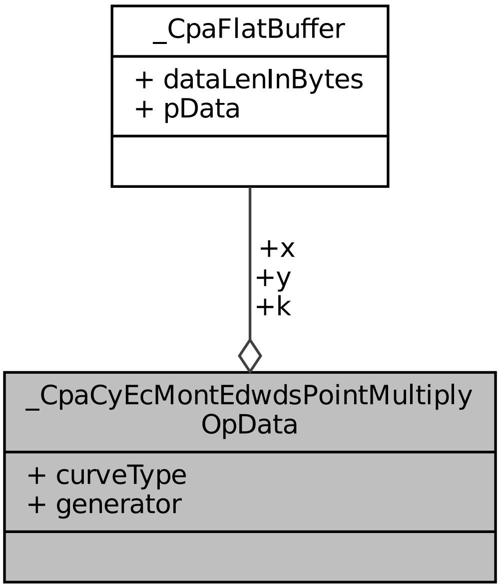
`CpaCyEcCurve* _CpaCyEcGenericPointVerifyOpData::pCurve`

<y coordinate of public key

Definition at line 416 of file cpa_cy_ec.h.

5.26 _CpaCyEcMontEdwdsPointMultiplyOpData Struct Reference

Collaboration diagram for _CpaCyEcMontEdwdsPointMultiplyOpData:



Data Fields

- CpaCyEcMontEdwdsCurveType curveType
- CpaBoolean generator
- CpaFlatBuffer k
- CpaFlatBuffer x
- CpaFlatBuffer y

5.26.1 Detailed Description

EC Point Multiplication Operation Data for Edwards or Montgomery curves as specificied in RFC#7748.

Description:

This structure contains the operation data for the cpaCyEcMontEdwdsPointMultiply function. The client MUST allocate the memory for this structure and the items pointed to by this structure. When the structure is passed into the function, ownership of the memory passes to the function. Ownership of the memory returns to the client when this structure is returned in the callback function.

For optimal performance all data buffers SHOULD be 8-byte aligned.

All values in this structure are required to be in Most Significant Byte first order, e.g. a.pData[0] = MSB.

Note

If the client modifies or frees the memory referenced in this structure after it has been submitted to the cpaCyEcMontEdwdsPointMultiply function, and before it has been returned in the callback, undefined behavior will result.

All buffers in this structure need to be:

- 32 bytes in size for 25519 curves
- 64 bytes in size for 448 curves

See also

cpaCyEcMontEdwdsPointMultiply()

Definition at line 454 of file cpa_cy_ec.h.

5.26.2 Field Documentation

5.26.2.1 curveType

`CpaCyEcMontEdwdsCurveType _CpaCyEcMontEdwdsPointMultiplyOpData::curveType`

field type for the operation

Definition at line 455 of file cpa_cy_ec.h.

5.26.2.2 generator

`CpaBoolean _CpaCyEcMontEdwdsPointMultiplyOpData::generator`

True if the operation is a generator multiplication (kG) False if it is a variable point multiplcation (kP).

Definition at line 457 of file cpa_cy_ec.h.

5.26.2.3 k

`CpaFlatBuffer _CpaCyEcMontEdwdsPointMultiplyOpData::k`

k scalar multiplier for the operation

Definition at line 460 of file cpa_cy_ec.h.

5.26.2.4 x

`CpaFlatBuffer _CpaCyEcMontEdwdsPointMultiplyOpData::x`

x value. Used in scalar varable point multiplication operations. Not required if the generator is True. Must be NULL if not required. The size of the buffer MUST be 32B for 25519 curves and 64B for 448 curves

Definition at line 462 of file cpa_cy_ec.h.

5.26.2.5 y

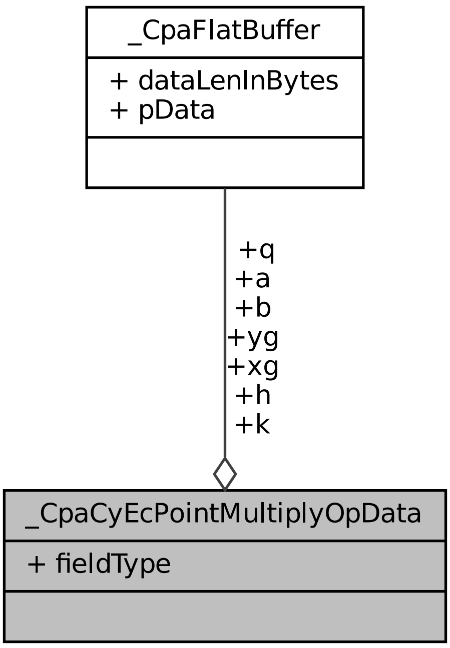
`CpaFlatBuffer _CpaCyEcMontEdwdsPointMultiplyOpData::y`

y value. Used in variable point multiplication of operations. Not required if the generator is True. Must be NULL if not required. The size of the buffer MUST be 32B for 25519 curves and 64B for 448 curves

Definition at line 467 of file cpa_cy_ec.h.

5.27 _CpaCyEcPointMultiplyOpData Struct Reference

Collaboration diagram for _CpaCyEcPointMultiplyOpData:



Data Fields

- CpaFlatBuffer k
- CpaFlatBuffer xg
- CpaFlatBuffer yg
- CpaFlatBuffer a
- CpaFlatBuffer b
- CpaFlatBuffer q
- CpaFlatBuffer h
- CpaCyEcFieldType fieldType

5.27.1 Detailed Description

EC Point Multiplication Operation Data.

Description:

This structure contains the operation data for the cpaCyEcPointMultiply function. The client MUST allocate the memory for this structure and the items pointed to by this structure. When the structure is passed into the function, ownership of the memory passes to the function. Ownership of the memory returns to the client when this structure is returned in the callback function.

For optimal performance all data buffers SHOULD be 8-byte aligned.

All values in this structure are required to be in Most Significant Byte first order, e.g. a.pData[0] = MSB.

Note

If the client modifies or frees the memory referenced in this structure after it has been submitted to the cpaCyEcPointMultiply function, and before it has been returned in the callback, undefined behavior will result.

See also

cpaCyEcPointMultiply()

Definition at line 317 of file cpa_cy_ec.h.

5.27.2 Field Documentation

5.27.2.1 k

`CpaFlatBuffer _CpaCyEcPointMultiplyOpData::k`

scalar multiplier ($k > 0$ and $k < n$)

Definition at line 318 of file cpa_cy_ec.h.

5.27.2.2 xg

`CpaFlatBuffer _CpaCyEcPointMultiplyOpData::xg`

x coordinate of curve point

Definition at line 320 of file cpa_cy_ec.h.

5.27.2.3 yg

`CpaFlatBuffer _CpaCyEcPointMultiplyOpData::yg`

y coordinate of curve point

Definition at line 322 of file cpa_cy_ec.h.

5.27.2.4 a

`CpaFlatBuffer _CpaCyEcPointMultiplyOpData::a`

a elliptic curve coefficient

Definition at line 324 of file cpa_cy_ec.h.

5.27.2.5 b

`CpaFlatBuffer _CpaCyEcPointMultiplyOpData::b`

b elliptic curve coefficient

Definition at line 326 of file cpa_cy_ec.h.

5.27.2.6 q

`CpaFlatBuffer _CpaCyEcPointMultiplyOpData::q`

prime modulus or irreducible polynomial over GF(2^m)

Definition at line 328 of file cpa_cy_ec.h.

5.27.2.7 h

`CpaFlatBuffer _CpaCyEcPointMultiplyOpData::h`

cofactor of the operation. If the cofactor is NOT required then set the cofactor to 1 or the data pointer of the Flat Buffer to NULL.

Definition at line 330 of file cpa_cy_ec.h.

5.27.2.8 fieldType

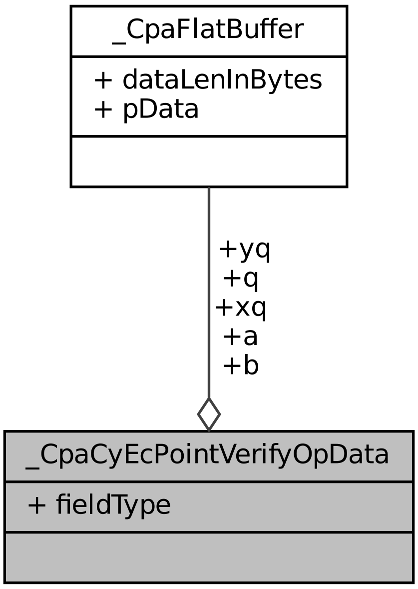
`CpaCyEcFieldType _CpaCyEcPointMultiplyOpData::fieldType`

field type for the operation

Definition at line 334 of file cpa_cy_ec.h.

5.28 _CpaCyEcPointVerifyOpData Struct Reference

Collaboration diagram for _CpaCyEcPointVerifyOpData:



Data Fields

- CpaFlatBuffer xq
- CpaFlatBuffer yq
- CpaFlatBuffer q
- CpaFlatBuffer a
- CpaFlatBuffer b
- CpaCyEcFieldType fieldType

5.28.1 Detailed Description

EC Point Verification Operation Data.

Description:

This structure contains the operation data for the cpaCyEcPointVerify function. The client MUST allocate the memory for this structure and the items pointed to by this structure. When the structure is passed into the function, ownership of the memory passes to the function. Ownership of the memory returns to the client when this structure is returned in the callback function.

For optimal performance all data buffers SHOULD be 8-byte aligned.

All values in this structure are required to be in Most Significant Byte first order, e.g. a.pData[0] = MSB.

Note

If the client modifies or frees the memory referenced in this structure after it has been submitted to the CpaCyEcPointVerify function, and before it has been returned in the callback, undefined behavior will result.

See also

cpaCyEcPointVerify()

Definition at line 503 of file cpa_cy_ec.h.

5.28.2 Field Documentation

5.28.2.1 xq

```
CpaFlatBuffer _CpaCyEcPointVerifyOpData::xq
```

x coordinate candidate point

Definition at line 504 of file cpa_cy_ec.h.

5.28.2.2 yq

`CpaFlatBuffer _CpaCyEcPointVerifyOpData::yq`

y coordinate candidate point

Definition at line 506 of file cpa_cy_ec.h.

5.28.2.3 q

`CpaFlatBuffer _CpaCyEcPointVerifyOpData::q`

prime modulus or irreducible polynomial over GF(2^m)

Definition at line 508 of file cpa_cy_ec.h.

5.28.2.4 a

`CpaFlatBuffer _CpaCyEcPointVerifyOpData::a`

a elliptic curve coefficient

Definition at line 510 of file cpa_cy_ec.h.

5.28.2.5 b

`CpaFlatBuffer _CpaCyEcPointVerifyOpData::b`

b elliptic curve coefficient

Definition at line 512 of file cpa_cy_ec.h.

5.28.2.6 fieldType

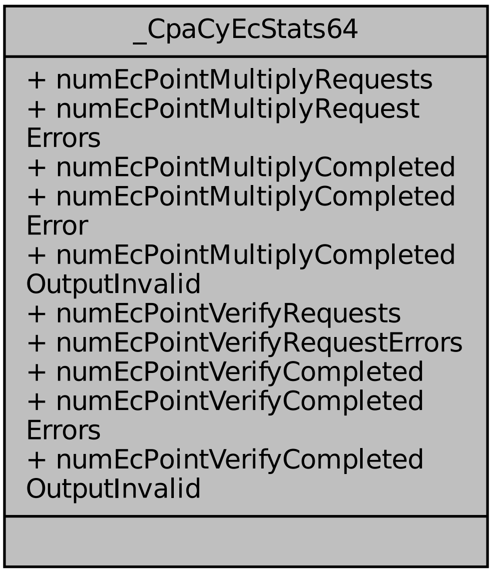
`CpaCyEcFieldType _CpaCyEcPointVerifyOpData::fieldType`

field type for the operation

Definition at line 514 of file cpa_cy_ec.h.

5.29 _CpaCyEcStats64 Struct Reference

Collaboration diagram for _CpaCyEcStats64:



Data Fields

- Cpa64U numEcPointMultiplyRequests
- Cpa64U numEcPointMultiplyRequestErrors
- Cpa64U numEcPointMultiplyCompleted
- Cpa64U numEcPointMultiplyCompletedError
- Cpa64U numEcPointMultiplyCompletedOutputInvalid
- Cpa64U numEcPointVerifyRequests
- Cpa64U numEcPointVerifyRequestErrors
- Cpa64U numEcPointVerifyCompleted
- Cpa64U numEcPointVerifyCompletedErrors
- Cpa64U numEcPointVerifyCompletedOutputInvalid

5.29.1 Detailed Description

Cryptographic EC Statistics.

Description:

This structure contains statistics on the Cryptographic EC operations. Statistics are set to zero when the component is initialized, and are collected per instance.

Definition at line 529 of file cpa_cy_ec.h.

5.29.2 Field Documentation

5.29.2.1 numEcPointMultiplyRequests

`Cpa64U _CpaCyEcStats64::numEcPointMultiplyRequests`

Total number of EC Point Multiplication operation requests.

Definition at line 530 of file cpa_cy_ec.h.

5.29.2.2 numEcPointMultiplyRequestErrors

`Cpa64U _CpaCyEcStats64::numEcPointMultiplyRequestErrors`

Total number of EC Point Multiplication operation requests that had an error and could not be processed.

Definition at line 532 of file cpa_cy_ec.h.

5.29.2.3 numEcPointMultiplyCompleted

`Cpa64U _CpaCyEcStats64::numEcPointMultiplyCompleted`

Total number of EC Point Multiplication operation requests that completed successfully.

Definition at line 535 of file cpa_cy_ec.h.

5.29.2.4 numEcPointMultiplyCompletedError

`Cpa64U _CpaCyEcStats64::numEcPointMultiplyCompletedError`

Total number of EC Point Multiplication operation requests that could not be completed successfully due to errors.

Definition at line 538 of file cpa_cy_ec.h.

5.29.2.5 numEcPointMultiplyCompletedOutputInvalid

`Cpa64U _CpaCyEcStats64::numEcPointMultiplyCompletedOutputInvalid`

Total number of EC Point Multiplication operation requests that could not be completed successfully due to an invalid output. Note that this does not indicate an error.

Definition at line 541 of file cpa_cy_ec.h.

5.29.2.6 numEcPointVerifyRequests

`Cpa64U _CpaCyEcStats64::numEcPointVerifyRequests`

Total number of EC Point Verification operation requests.

Definition at line 545 of file cpa_cy_ec.h.

5.29.2.7 numEcPointVerifyRequestErrors

`Cpa64U _CpaCyEcStats64::numEcPointVerifyRequestErrors`

Total number of EC Point Verification operation requests that had an error and could not be processed.

Definition at line 547 of file cpa_cy_ec.h.

5.29.2.8 numEcPointVerifyCompleted

`Cpa64U _CpaCyEcStats64::numEcPointVerifyCompleted`

Total number of EC Point Verification operation requests that completed successfully.

Definition at line 550 of file cpa_cy_ec.h.

5.29.2.9 numEcPointVerifyCompletedErrors

`Cpa64U _CpaCyEcStats64::numEcPointVerifyCompletedErrors`

Total number of EC Point Verification operation requests that could not be completed successfully due to errors.

Definition at line 553 of file cpa_cy_ec.h.

5.29.2.10 numEcPointVerifyCompletedOutputInvalid

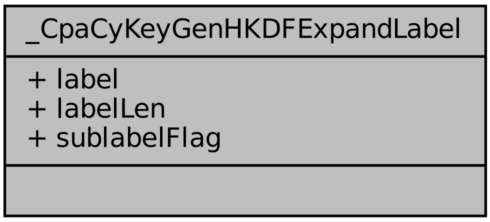
`Cpa64U _CpaCyEcStats64::numEcPointVerifyCompletedOutputInvalid`

Total number of EC Point Verification operation requests that had an invalid output. Note that this does not indicate an error.

Definition at line 556 of file cpa_cy_ec.h.

5.30 _CpaCyKeyGenHKDFExpandLabel Struct Reference

Collaboration diagram for _CpaCyKeyGenHKDFExpandLabel:



Data Fields

- Cpa8U label [CPA_CY_HKDF_KEY_MAX_LABEL_SZ]
- Cpa8U labelLen
- Cpa8U sublabelFlag

5.30.1 Detailed Description

Maximum number of labels in op structure

File: cpa_cy_key.h

TLS data for key generation functions

Description:

This structure contains data for describing label for the HKDF Extract Label function

Extract Label Function

labelLen = length of the label field
contextLen = length of the context field
sublabelFlag = Mask of sub labels required for this label.
label = label as defined in RFC8446
context = context as defined in RFC8446

Definition at line 299 of file cpa_cy_key.h.

5.30.2 Field Documentation

5.30.2.1 label

`Cpa8U _CpaCyKeyGenHKDFExpandLabel::label[CPA_CY_HKDF_KEY_MAX_LABEL_SZ]`

HKDFLabel field as defined in RFC8446 sec 7.1.

Definition at line 301 of file cpa_cy_key.h.

5.30.2.2 labelLen

`Cpa8U _CpaCyKeyGenHKDFExpandLabel::labelLen`

The length, in bytes of the label

Definition at line 304 of file cpa_cy_key.h.

5.30.2.3 sublabelFlag

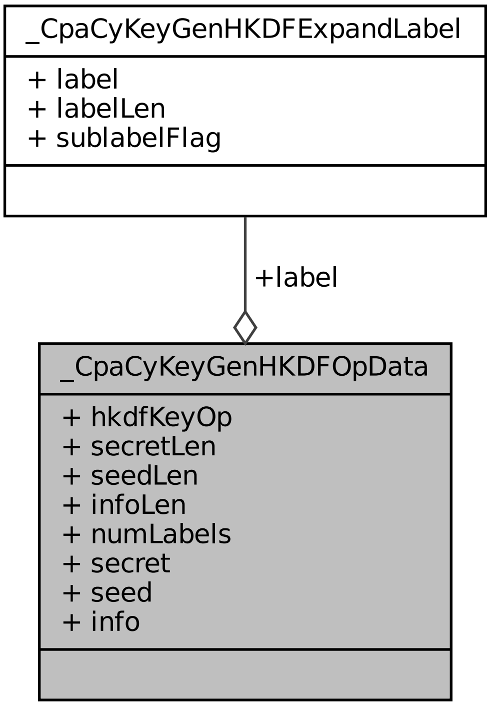
`Cpa8U _CpaCyKeyGenHKDFExpandLabel::sublabelFlag`

mask of sublabels to be generated. This flag is composed of zero or more of: CPA_CY_HKDF_SUBLABEL_KEY CPA_CY_HKDF_SUBLABEL_IV CPA_CY_HKDF_SUBLABEL_RESUMPTION CPA_CY_HKDF_SUBLABEL_FINISHED

Definition at line 306 of file cpa_cy_key.h.

5.31 _CpaCyKeyGenHKDFOpData Struct Reference

Collaboration diagram for _CpaCyKeyGenHKDFOpData:



Data Fields

- CpaCyKeyHKDFOp hkdfKeyOp
- Cpa8U secretLen
- Cpa16U seedLen
- Cpa16U infoLen
- Cpa16U numLabels
- Cpa8U secret [CPA_CY_HKDF_KEY_MAX_SECRET_SZ]
- Cpa8U seed [CPA_CY_HKDF_KEY_MAX_HMAC_SZ]
- Cpa8U info [CPA_CY_HKDF_KEY_MAX_INFO_SZ]
- CpaCyKeyGenHKDFExpandLabel label [CPA_CY_HKDF_KEY_MAX_LABEL_COUNT]

5.31.1 Detailed Description

TLS data for key generation functions

Description:

This structure contains data for all HKDF operations:
HKDF Extract
HKDF Expand
HKDF Expand Label
HKDF Extract and Expand
HKDF Extract and Expand Label

HKDF Map Structure Elements

secret - IKM value for extract operations or PRK for expand or expand operations.
seed - contains the salt for extract operations
info - contains the info data for extract operations
labels - See notes above

Definition at line 337 of file cpa_cy_key.h.

5.31.2 Field Documentation

5.31.2.1 hkdfKeyOp

`CpaCyKeyHKDFOp _CpaCyKeyGenHKDFOpData::hkdfKeyOp`

Keying operation to be performed.

Definition at line 339 of file cpa_cy_key.h.

5.31.2.2 secretLen

`Cpa8U _CpaCyKeyGenHKDFOpData::secretLen`

Length of secret field

Definition at line 341 of file cpa_cy_key.h.

5.31.2.3 seedLen

`Cpa16U _CpaCyKeyGenHKDFOpData::seedLen`

Length of seed field

Definition at line 343 of file cpa_cy_key.h.

5.31.2.4 infoLen

```
Cpa16U _CpaCyKeyGenHKDFOpData::infoLen
```

Length of info field

Definition at line 345 of file cpa_cy_key.h.

5.31.2.5 numLabels

```
Cpa16U _CpaCyKeyGenHKDFOpData::numLabels
```

Number of filled CpaCyKeyGenHKDFExpandLabel elements

Definition at line 347 of file cpa_cy_key.h.

5.31.2.6 secret

```
Cpa8U _CpaCyKeyGenHKDFOpData::secret[CPA_CY_HKDF_KEY_MAX_SECRET_SZ]
```

Input Key Material or PRK

Definition at line 349 of file cpa_cy_key.h.

5.31.2.7 seed

```
Cpa8U _CpaCyKeyGenHKDFOpData::seed[CPA_CY_HKDF_KEY_MAX_HMAC_SZ]
```

Input salt

Definition at line 351 of file cpa_cy_key.h.

5.31.2.8 info

```
Cpa8U _CpaCyKeyGenHKDFOpData::info[CPA_CY_HKDF_KEY_MAX_INFO_SZ]
```

info field

Definition at line 353 of file cpa_cy_key.h.

5.31.2.9 label

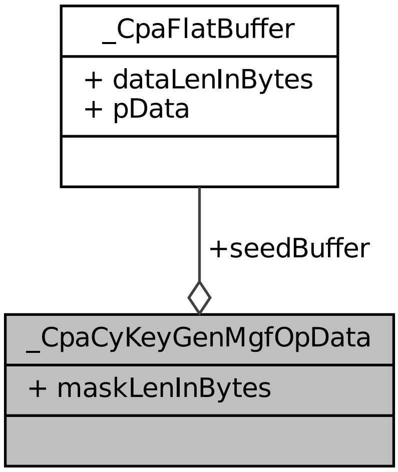
`CpaCyKeyGenHKDFExpandLabel _CpaCyKeyGenHKDFOpData::label[CPA_CY_HKDF_KEY_MAX_LABEL_COUNT]`

array of Expand Label structures

Definition at line 355 of file cpa_cy_key.h.

5.32 _CpaCyKeyGenMgfOpData Struct Reference

Collaboration diagram for _CpaCyKeyGenMgfOpData:



Data Fields

- CpaFlatBuffer seedBuffer
- Cpa32U maskLenInBytes

5.32.1 Detailed Description

Key Generation Mask Generation Function (MGF) Data

Description:

This structure contains data relating to Mask Generation Function key generation operations.

Note

The default hash algorithm used by the MGF is SHA-1. If a different hash algorithm is preferred, then see the extended version of this structure, CpaCyKeyGenMgfOpDataExt.

See also

cpaCyKeyGenMgf

Definition at line 433 of file cpa_cy_key.h.

5.32.2 Field Documentation

5.32.2.1 seedBuffer

`CpaFlatBuffer _CpaCyKeyGenMgfOpData::seedBuffer`

Caller MUST allocate a buffer and populate with the input seed data. For optimal performance the start of the seed SHOULD be allocated on an 8-byte boundary. The length field represents the seed length in bytes. Implementation-specific limits may apply to this length.

Definition at line 434 of file cpa_cy_key.h.

5.32.2.2 maskLenInBytes

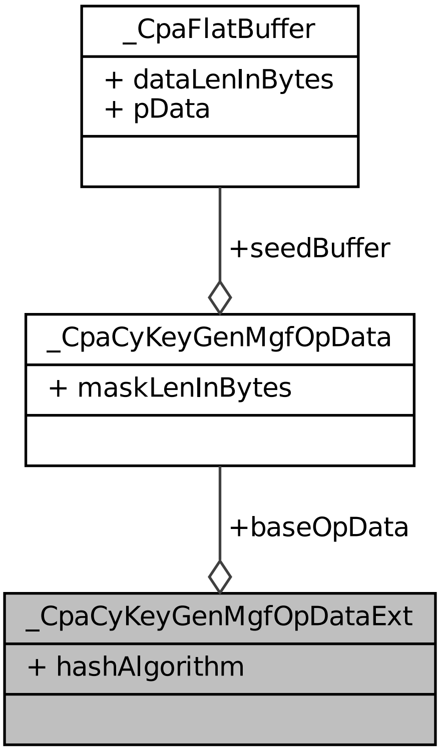
`Cpa32U _CpaCyKeyGenMgfOpData::maskLenInBytes`

The requested length of the generated mask in bytes. Implementation-specific limits may apply to this length.

Definition at line 439 of file cpa_cy_key.h.

5.33 _CpaCyKeyGenMgfOpDataExt Struct Reference

Collaboration diagram for _CpaCyKeyGenMgfOpDataExt:



Data Fields

- CpaCyKeyGenMgfOpData baseOpData
- CpaCySymHashAlgorithm hashAlgorithm

5.33.1 Detailed Description

Extension to the original Key Generation Mask Generation Function (MGF) Data

Description:

This structure is an extension to the original MGF data structure. The extension allows the hash function to be specified.

Note

This structure is separate from the base CpaCyKeyGenMgfOpData structure in order to retain backwards compatibility with the original version of the API.

See also

cpaCyKeyGenMgfExt

Definition at line 459 of file cpa_cy_key.h.

5.33.2 Field Documentation

5.33.2.1 baseOpData

`CpaCyKeyGenMgfOpData _CpaCyKeyGenMgfOpDataExt::baseOpData`

"Base" operational data for MGF generation

Definition at line 460 of file cpa_cy_key.h.

5.33.2.2 hashAlgorithm

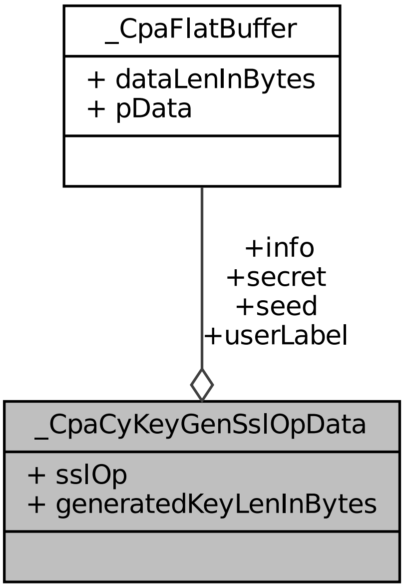
`CpaCySymHashAlgorithm _CpaCyKeyGenMgfOpDataExt::hashAlgorithm`

Specifies the hash algorithm to be used by the Mask Generation Function

Definition at line 462 of file cpa_cy_key.h.

5.34 _CpaCyKeyGenSslOpData Struct Reference

Collaboration diagram for _CpaCyKeyGenSslOpData:



Data Fields

- CpaCyKeySslOp sslOp
- CpaFlatBuffer secret
- CpaFlatBuffer seed
- CpaFlatBuffer info
- Cpa32U generatedKeyLenInBytes
- CpaFlatBuffer userLabel

5.34.1 Detailed Description

SSL data for key generation functions

Description:

This structure contains data for use in key generation operations for SSL. For specific SSL key generation operations, the structure fields MUST be set as follows:

SSL Master-Secret Derivation:

sslOp = CPA_CY_KEY_SSL_OP_MASTER_SECRET_DERIVE
secret = pre-master secret key
seed = client_random + server_random
userLabel = NULL

SSL Key-Material Derivation:

sslOp = CPA_CY_KEY_SSL_OP_KEY_MATERIAL_DERIVE
secret = master secret key
seed = server_random + client_random
userLabel = NULL

Note that the client/server random order is reversed from that used for master-secret derivation.

Note

Each of the client and server random numbers need to be of length CPA_CY_KEY_GEN_SSL_TLS_RANDOM_LEN_IN_BYTES.

In each of the above descriptions, + indicates concatenation.

The label used is predetermined by the SSL operation in line with the SSL 3.0 specification, and can be overridden by using a user defined operation CPA_CY_KEY_SSL_OP_USER_DEFINED and associated userLabel.

Definition at line 103 of file cpa_cy_key.h.

5.34.2 Field Documentation

5.34.2.1 sslOp

`CpaCyKeySslOp _CpaCyKeyGenSslOpData::sslOp`

Indicate the SSL operation to be performed

Definition at line 104 of file cpa_cy_key.h.

5.34.2.2 secret

`CpaFlatBuffer _CpaCyKeyGenSslOpData::secret`

Flat buffer containing a pointer to either the master or pre-master secret key. The length field indicates the length of the secret key in bytes. Implementation-specific limits may apply to this length.

Definition at line 106 of file cpa_cy_key.h.

5.34.2.3 seed

`CpaFlatBuffer _CpaCyKeyGenSslOpData::seed`

Flat buffer containing a pointer to the seed data. Implementation-specific limits may apply to this length.

Definition at line 110 of file cpa_cy_key.h.

5.34.2.4 info

`CpaFlatBuffer _CpaCyKeyGenSslOpData::info`

Flat buffer containing a pointer to the info data. Implementation-specific limits may apply to this length.

Definition at line 113 of file cpa_cy_key.h.

5.34.2.5 generatedKeyLenInBytes

`Cpa32U _CpaCyKeyGenSslOpData::generatedKeyLenInBytes`

The requested length of the generated key in bytes. Implementation-specific limits may apply to this length.

Definition at line 116 of file cpa_cy_key.h.

5.34.2.6 userLabel

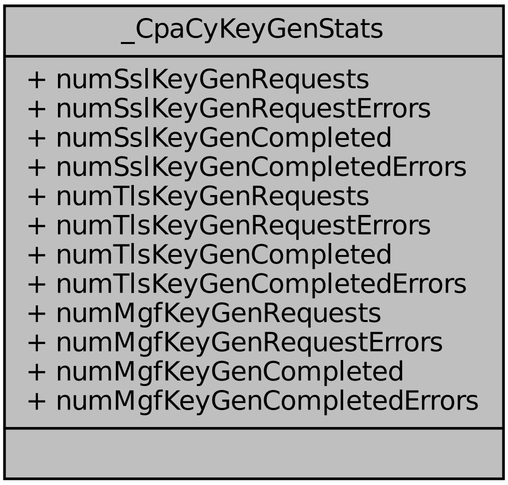
`CpaFlatBuffer _CpaCyKeyGenSslOpData::userLabel`

Optional flat buffer containing a pointer to a user defined label. The length field indicates the length of the label in bytes. To use this field, the sslOp must be CPA_CY_KEY_SSL_OP_USER_DEFINED, or otherwise it is ignored and can be set to NULL. Implementation-specific limits may apply to this length.

Definition at line 119 of file cpa_cy_key.h.

5.35 _CpaCyKeyGenStats Struct Reference

Collaboration diagram for _CpaCyKeyGenStats:



Data Fields

- Cpa32U numSslKeyGenRequests
- Cpa32U numSslKeyGenRequestErrors
- Cpa32U numSslKeyGenCompleted
- Cpa32U numSslKeyGenCompletedErrors
- Cpa32U numTlsKeyGenRequests
- Cpa32U numTlsKeyGenRequestErrors
- Cpa32U numTlsKeyGenCompleted
- Cpa32U numTlsKeyGenCompletedErrors
- Cpa32U numMgfKeyGenRequests
- Cpa32U numMgfKeyGenRequestErrors
- Cpa32U numMgfKeyGenCompleted
- Cpa32U numMgfKeyGenCompletedErrors

5.35.1 Detailed Description

Key Generation Statistics.

Deprecated As of v1.3 of the Crypto API, this structure has been deprecated, replaced by CpaCyKeyGenStats64.

Description:

This structure contains statistics on the key and mask generation operations. Statistics are set to zero when the component is initialized, and are collected per instance.

Definition at line 480 of file cpa_cy_key.h.

5.35.2 Field Documentation

5.35.2.1 numSslKeyGenRequests

`Cpa32U _CpaCyKeyGenStats::numSslKeyGenRequests`

Total number of successful SSL key generation requests.

Definition at line 481 of file cpa_cy_key.h.

5.35.2.2 numSslKeyGenRequestErrors

`Cpa32U _CpaCyKeyGenStats::numSslKeyGenRequestErrors`

Total number of SSL key generation requests that had an error and could not be processed.

Definition at line 483 of file cpa_cy_key.h.

5.35.2.3 numSslKeyGenCompleted

`Cpa32U _CpaCyKeyGenStats::numSslKeyGenCompleted`

Total number of SSL key generation operations that completed successfully.

Definition at line 486 of file cpa_cy_key.h.

5.35.2.4 numSslKeyGenCompletedErrors

`Cpa32U _CpaCyKeyGenStats::numSslKeyGenCompletedErrors`

Total number of SSL key generation operations that could not be completed successfully due to errors.

Definition at line 489 of file cpa_cy_key.h.

5.35.2.5 numTlsKeyGenRequests

`Cpa32U _CpaCyKeyGenStats::numTlsKeyGenRequests`

Total number of successful TLS key generation requests.

Definition at line 492 of file cpa_cy_key.h.

5.35.2.6 numTlsKeyGenRequestErrors

`Cpa32U _CpaCyKeyGenStats::numTlsKeyGenRequestErrors`

Total number of TLS key generation requests that had an error and could not be processed.

Definition at line 494 of file cpa_cy_key.h.

5.35.2.7 numTlsKeyGenCompleted

`Cpa32U _CpaCyKeyGenStats::numTlsKeyGenCompleted`

Total number of TLS key generation operations that completed successfully.

Definition at line 497 of file cpa_cy_key.h.

5.35.2.8 numTlsKeyGenCompletedErrors

`Cpa32U _CpaCyKeyGenStats::numTlsKeyGenCompletedErrors`

Total number of TLS key generation operations that could not be completed successfully due to errors.

Definition at line 500 of file cpa_cy_key.h.

5.35.2.9 numMgfKeyGenRequests

`Cpa32U _CpaCyKeyGenStats::numMgfKeyGenRequests`

Total number of successful MGF key generation requests (including "extended" MGF requests).

Definition at line 503 of file cpa_cy_key.h.

5.35.2.10 numMgfKeyGenRequestErrors

`Cpa32U _CpaCyKeyGenStats::numMgfKeyGenRequestErrors`

Total number of MGF key generation requests that had an error and could not be processed.

Definition at line 506 of file cpa_cy_key.h.

5.35.2.11 numMgfKeyGenCompleted

`Cpa32U _CpaCyKeyGenStats::numMgfKeyGenCompleted`

Total number of MGF key generation operations that completed successfully.

Definition at line 509 of file cpa_cy_key.h.

5.35.2.12 numMgfKeyGenCompletedErrors

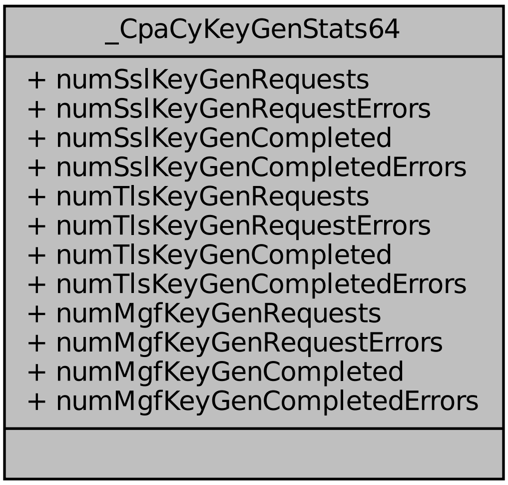
`Cpa32U _CpaCyKeyGenStats::numMgfKeyGenCompletedErrors`

Total number of MGF key generation operations that could not be completed successfully due to errors.

Definition at line 512 of file cpa_cy_key.h.

5.36 _CpaCyKeyGenStats64 Struct Reference

Collaboration diagram for _CpaCyKeyGenStats64:



Data Fields

- Cpa64U numSslKeyGenRequests
- Cpa64U numSslKeyGenRequestErrors
- Cpa64U numSslKeyGenCompleted
- Cpa64U numSslKeyGenCompletedErrors
- Cpa64U numTlsKeyGenRequests
- Cpa64U numTlsKeyGenRequestErrors
- Cpa64U numTlsKeyGenCompleted
- Cpa64U numTlsKeyGenCompletedErrors
- Cpa64U numMgfKeyGenRequests
- Cpa64U numMgfKeyGenRequestErrors
- Cpa64U numMgfKeyGenCompleted
- Cpa64U numMgfKeyGenCompletedErrors

5.36.1 Detailed Description

Key Generation Statistics (64-bit version).

Description:

This structure contains the 64-bit version of the statistics on the key and mask generation operations. Statistics are set to zero when the component is initialized, and are collected per instance.

Definition at line 528 of file cpa_cy_key.h.

5.36.2 Field Documentation

5.36.2.1 numSslKeyGenRequests

`Cpa64U _CpaCyKeyGenStats64::numSslKeyGenRequests`

Total number of successful SSL key generation requests.

Definition at line 529 of file cpa_cy_key.h.

5.36.2.2 numSslKeyGenRequestErrors

`Cpa64U _CpaCyKeyGenStats64::numSslKeyGenRequestErrors`

Total number of SSL key generation requests that had an error and could not be processed.

Definition at line 531 of file cpa_cy_key.h.

5.36.2.3 numSslKeyGenCompleted

`Cpa64U _CpaCyKeyGenStats64::numSslKeyGenCompleted`

Total number of SSL key generation operations that completed successfully.

Definition at line 534 of file cpa_cy_key.h.

5.36.2.4 numSslKeyGenCompletedErrors

```
Cpa64U _CpaCyKeyGenStats64::numSslKeyGenCompletedErrors
```

Total number of SSL key generation operations that could not be completed successfully due to errors.

Definition at line 537 of file cpa_cy_key.h.

5.36.2.5 numTlsKeyGenRequests

```
Cpa64U _CpaCyKeyGenStats64::numTlsKeyGenRequests
```

Total number of successful TLS key generation requests.

Definition at line 540 of file cpa_cy_key.h.

5.36.2.6 numTlsKeyGenRequestErrors

```
Cpa64U _CpaCyKeyGenStats64::numTlsKeyGenRequestErrors
```

Total number of TLS key generation requests that had an error and could not be processed.

Definition at line 542 of file cpa_cy_key.h.

5.36.2.7 numTlsKeyGenCompleted

```
Cpa64U _CpaCyKeyGenStats64::numTlsKeyGenCompleted
```

Total number of TLS key generation operations that completed successfully.

Definition at line 545 of file cpa_cy_key.h.

5.36.2.8 numTlsKeyGenCompletedErrors

```
Cpa64U _CpaCyKeyGenStats64::numTlsKeyGenCompletedErrors
```

Total number of TLS key generation operations that could not be completed successfully due to errors.

Definition at line 548 of file cpa_cy_key.h.

5.36.2.9 numMgfKeyGenRequests

`Cpa64U _CpaCyKeyGenStats64::numMgfKeyGenRequests`

Total number of successful MGF key generation requests (including "extended" MGF requests).

Definition at line 551 of file cpa_cy_key.h.

5.36.2.10 numMgfKeyGenRequestErrors

`Cpa64U _CpaCyKeyGenStats64::numMgfKeyGenRequestErrors`

Total number of MGF key generation requests that had an error and could not be processed.

Definition at line 554 of file cpa_cy_key.h.

5.36.2.11 numMgfKeyGenCompleted

`Cpa64U _CpaCyKeyGenStats64::numMgfKeyGenCompleted`

Total number of MGF key generation operations that completed successfully.

Definition at line 557 of file cpa_cy_key.h.

5.36.2.12 numMgfKeyGenCompletedErrors

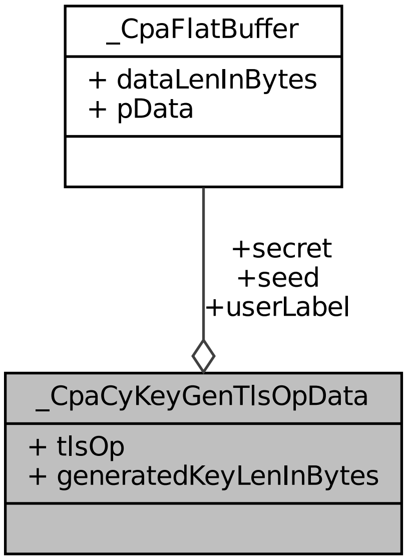
`Cpa64U _CpaCyKeyGenStats64::numMgfKeyGenCompletedErrors`

Total number of MGF key generation operations that could not be completed successfully due to errors.

Definition at line 560 of file cpa_cy_key.h.

5.37 _CpaCyKeyGenTlsOpData Struct Reference

Collaboration diagram for _CpaCyKeyGenTlsOpData:



Data Fields

- CpaCyKeyTlsOp tlsOp
- CpaFlatBuffer secret
- CpaFlatBuffer seed
- Cpa32U generatedKeyLenInBytes
- CpaFlatBuffer userLabel

5.37.1 Detailed Description

TLS data for key generation functions

Description:

This structure contains data for use in key generation operations for TLS. For specific TLS key generation operations, the structure fields MUST be set as follows:

TLS Master-Secret Derivation:

tlsOp = CPA_CY_KEY_TLS_OP_MASTER_SECRET_DERIVE
secret = pre-master secret key
seed = client_random + server_random
userLabel = NULL

TLS Key-Material Derivation:

tlsOp = CPA_CY_KEY_TLS_OP_KEY_MATERIAL_DERIVE
secret = master secret key
seed = server_random + client_random
userLabel = NULL

Note that the client/server random order is reversed from that used for Master-Secret Derivation.

TLS Client finished/Server finished tag Derivation:

tlsOp = CPA_CY_KEY_TLS_OP_CLIENT_FINISHED_DERIVE (client)
or CPA_CY_KEY_TLS_OP_SERVER_FINISHED_DERIVE (server)
secret = master secret key
seed = MD5(handshake_messages) + SHA-1(handshake_messages)
userLabel = NULL

Note

Each of the client and server random seeds need to be of length CPA_CY_KEY_GEN_SSL_TLS_RANDOM_LEN_IN_BYTES.

In each of the above descriptions, + indicates concatenation.

The label used is predetermined by the TLS operation in line with the TLS specifications, and can be overridden by using a user defined operation CPA_CY_KEY_TLS_OP_USER_DEFINED and associated userLabel.

Definition at line 399 of file cpa_cy_key.h.

5.37.2 Field Documentation

5.37.2.1 tlsOp

`CpaCyKeyTlsOp _CpaCyKeyGenTlsOpData::tlsOp`

TLS operation to be performed

Definition at line 400 of file cpa_cy_key.h.

5.37.2.2 secret

`CpaFlatBuffer _CpaCyKeyGenTlsOpData::secret`

Flat buffer containing a pointer to either the master or pre-master secret key. The length field indicates the length of the secret in bytes.

Definition at line 402 of file cpa_cy_key.h.

5.37.2.3 seed

`CpaFlatBuffer _CpaCyKeyGenTlsOpData::seed`

Flat buffer containing a pointer to the seed data. Implementation-specific limits may apply to this length.

Definition at line 406 of file cpa_cy_key.h.

5.37.2.4 generatedKeyLenInBytes

`Cpa32U _CpaCyKeyGenTlsOpData::generatedKeyLenInBytes`

The requested length of the generated key in bytes. Implementation-specific limits may apply to this length.

Definition at line 409 of file cpa_cy_key.h.

5.37.2.5 userLabel

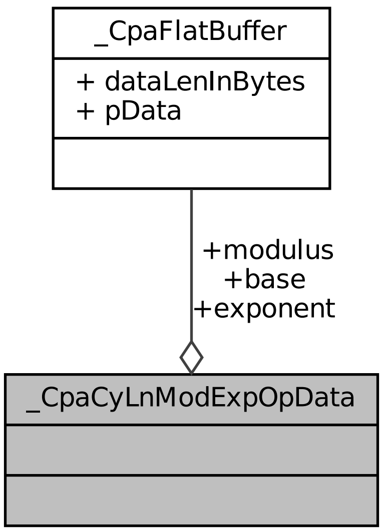
`CpaFlatBuffer _CpaCyKeyGenTlsOpData::userLabel`

Optional flat buffer containing a pointer to a user defined label. The length field indicates the length of the label in bytes. To use this field, the tlsOp must be CPA_CY_KEY_TLS_OP_USER_DEFINED. Implementation-specific limits may apply to this length.

Definition at line 412 of file cpa_cy_key.h.

5.38 _CpaCyLnModExpOpData Struct Reference

Collaboration diagram for _CpaCyLnModExpOpData:



Data Fields

- CpaFlatBuffer modulus
- CpaFlatBuffer base
- CpaFlatBuffer exponent

5.38.1 Detailed Description

Modular Exponentiation Function Operation Data.

Description:

This structure lists the different items that are required in the cpaCyLnModExp function. The client MUST allocate the memory for this structure. When the structure is passed into the function, ownership of the memory passes to the function. Ownership of the memory returns to the client when this structure is returned in the callback. The operation size in bits is equal to the size of whichever of the following is largest: the modulus, the base or the exponent.

Note

If the client modifies or frees the memory referenced in this structure after it has been submitted to the cpaCyLnModExp function, and before it has been returned in the callback, undefined behavior will result.

The values of the base, the exponent and the modulus MUST all be less than 2^8192, and the modulus must not be equal to zero.

Definition at line 85 of file cpa_cy_ln.h.

5.38.2 Field Documentation

5.38.2.1 modulus

`CpaFlatBuffer _CpaCyLnModExpOpData::modulus`

Flat buffer containing a pointer to the modulus. This number may be up to 8192 bits in length, and MUST be greater than zero.

Definition at line 86 of file cpa_cy_ln.h.

5.38.2.2 base

`CpaFlatBuffer _CpaCyLnModExpOpData::base`

Flat buffer containing a pointer to the base. This number may be up to 8192 bits in length.

Definition at line 91 of file cpa_cy_ln.h.

5.38.2.3 exponent

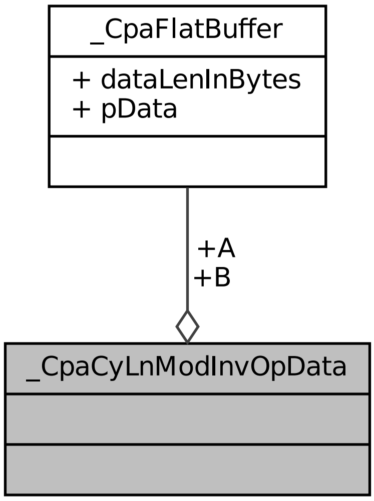
`CpaFlatBuffer _CpaCyLnModExpOpData::exponent`

Flat buffer containing a pointer to the exponent. This number may be up to 8192 bits in length.

Definition at line 95 of file cpa_cy_ln.h.

5.39 _CpaCyLnModInvOpData Struct Reference

Collaboration diagram for _CpaCyLnModInvOpData:



Data Fields

- CpaFlatBuffer A
- CpaFlatBuffer B

5.39.1 Detailed Description

Modular Inversion Function Operation Data.

Description:

This structure lists the different items that are required in the function cpaCyLnModInv. The client MUST allocate the memory for this structure. When the structure is passed into the function, ownership of the memory passes to the function. Ownership of the memory returns to the client when this structure is returned in the callback.

Note

If the client modifies or frees the memory referenced in this structure after it has been submitted to the cpaCyLnModInv function, and before it has been returned in the callback, undefined behavior will result.

Note that the values of A and B MUST NOT both be even numbers, and both MUST be less than 2^{8192}.

Definition at line 121 of file cpa_cy_ln.h.

5.39.2 Field Documentation

5.39.2.1 A

`CpaFlatBuffer _CpaCyLnModInvOpData::A`

Flat buffer containing a pointer to the value that will be inverted. This number may be up to 8192 bits in length, it MUST NOT be zero, and it MUST be co-prime with B.

Definition at line 122 of file cpa_cy_ln.h.

5.39.2.2 B

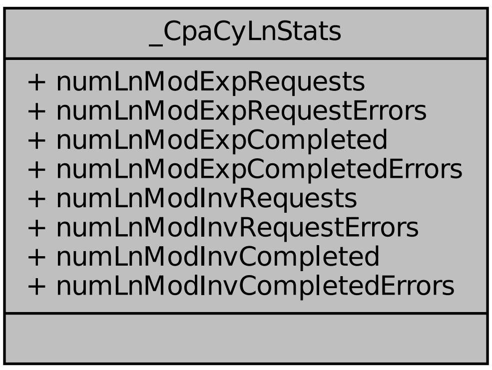
`CpaFlatBuffer _CpaCyLnModInvOpData::B`

Flat buffer containing a pointer to the value that will be used as the modulus. This number may be up to 8192 bits in length, it MUST NOT be zero, and it MUST be co-prime with A.

Definition at line 128 of file cpa_cy_ln.h.

5.40 _CpaCyLnStats Struct Reference

Collaboration diagram for _CpaCyLnStats:



Data Fields

- Cpa32U numLnModExpRequests
- Cpa32U numLnModExpRequestErrors
- Cpa32U numLnModExpCompleted
- Cpa32U numLnModExpCompletedErrors
- Cpa32U numLnModInvRequests
- Cpa32U numLnModInvRequestErrors
- Cpa32U numLnModInvCompleted
- Cpa32U numLnModInvCompletedErrors

5.40.1 Detailed Description

Look Aside Cryptographic large number Statistics.

Deprecated As of v1.3 of the Crypto API, this structure has been deprecated, replaced by CpaCyLnStats64.

Description:

This structure contains statistics on the Look Aside Cryptographic large number operations. Statistics are set to zero when the component is initialized, and are collected per instance.

Definition at line 149 of file cpa_cy_ln.h.

5.40.2 Field Documentation

5.40.2.1 numLnModExpRequests

`Cpa32U _CpaCyLnStats::numLnModExpRequests`

Total number of successful large number modular exponentiation requests.

Definition at line 150 of file cpa_cy_ln.h.

5.40.2.2 numLnModExpRequestErrors

`Cpa32U _CpaCyLnStats::numLnModExpRequestErrors`

Total number of large number modular exponentiation requests that had an error and could not be processed.

Definition at line 153 of file cpa_cy_ln.h.

5.40.2.3 numLnModExpCompleted

`Cpa32U _CpaCyLnStats::numLnModExpCompleted`

Total number of large number modular exponentiation operations that completed successfully.

Definition at line 156 of file cpa_cy_ln.h.

5.40.2.4 numLnModExpCompletedErrors

`Cpa32U _CpaCyLnStats::numLnModExpCompletedErrors`

Total number of large number modular exponentiation operations that could not be completed successfully due to errors.

Definition at line 159 of file cpa_cy_ln.h.

5.40.2.5 numLnModInvRequests

`Cpa32U _CpaCyLnStats::numLnModInvRequests`

Total number of successful large number modular inversion requests.

Definition at line 162 of file cpa_cy_ln.h.

5.40.2.6 numLnModInvRequestErrors

`Cpa32U _CpaCyLnStats::numLnModInvRequestErrors`

Total number of large number modular inversion requests that had an error and could not be processed.

Definition at line 165 of file cpa_cy_ln.h.

5.40.2.7 numLnModInvCompleted

`Cpa32U _CpaCyLnStats::numLnModInvCompleted`

Total number of large number modular inversion operations that completed successfully.

Definition at line 168 of file cpa_cy_ln.h.

5.40.2.8 numLnModInvCompletedErrors

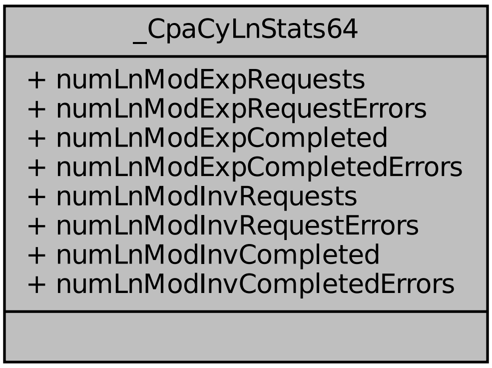
`Cpa32U _CpaCyLnStats::numLnModInvCompletedErrors`

Total number of large number modular inversion operations that could not be completed successfully due to errors.

Definition at line 171 of file cpa_cy_ln.h.

5.41 _CpaCyLnStats64 Struct Reference

Collaboration diagram for _CpaCyLnStats64:



Data Fields

- Cpa64U numLnModExpRequests
- Cpa64U numLnModExpRequestErrors
- Cpa64U numLnModExpCompleted
- Cpa64U numLnModExpCompletedErrors
- Cpa64U numLnModInvRequests
- Cpa64U numLnModInvRequestErrors
- Cpa64U numLnModInvCompleted
- Cpa64U numLnModInvCompletedErrors

5.41.1 Detailed Description

Look Aside Cryptographic large number Statistics.

Description:

This structure contains statistics on the Look Aside Cryptographic large number operations. Statistics are set to zero when the component is initialized, and are collected per instance.

Definition at line 186 of file cpa_cy_ln.h.

5.41.2 Field Documentation

5.41.2.1 numLnModExpRequests

`Cpa64U _CpaCyLnStats64::numLnModExpRequests`

Total number of successful large number modular exponentiation requests.

Definition at line 187 of file cpa_cy_ln.h.

5.41.2.2 numLnModExpRequestErrors

`Cpa64U _CpaCyLnStats64::numLnModExpRequestErrors`

Total number of large number modular exponentiation requests that had an error and could not be processed.

Definition at line 190 of file cpa_cy_ln.h.

5.41.2.3 numLnModExpCompleted

`Cpa64U _CpaCyLnStats64::numLnModExpCompleted`

Total number of large number modular exponentiation operations that completed successfully.

Definition at line 193 of file cpa_cy_ln.h.

5.41.2.4 numLnModExpCompletedErrors

`Cpa64U _CpaCyLnStats64::numLnModExpCompletedErrors`

Total number of large number modular exponentiation operations that could not be completed successfully due to errors.

Definition at line 196 of file cpa_cy_ln.h.

5.41.2.5 numLnModInvRequests

`Cpa64U _CpaCyLnStats64::numLnModInvRequests`

Total number of successful large number modular inversion requests.

Definition at line 199 of file cpa_cy_ln.h.

5.41.2.6 numLnModInvRequestErrors

`Cpa64U _CpaCyLnStats64::numLnModInvRequestErrors`

Total number of large number modular inversion requests that had an error and could not be processed.

Definition at line 202 of file cpa_cy_ln.h.

5.41.2.7 numLnModInvCompleted

`Cpa64U _CpaCyLnStats64::numLnModInvCompleted`

Total number of large number modular inversion operations that completed successfully.

Definition at line 205 of file cpa_cy_ln.h.

5.41.2.8 numLnModInvCompletedErrors

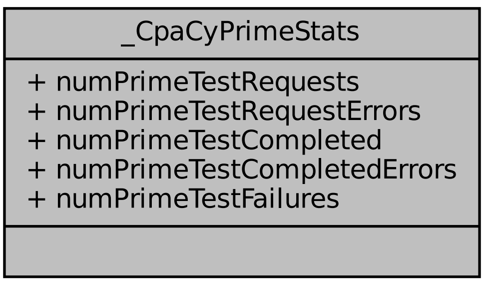
`Cpa64U _CpaCyLnStats64::numLnModInvCompletedErrors`

Total number of large number modular inversion operations that could not be completed successfully due to errors.

Definition at line 208 of file cpa_cy_ln.h.

5.42 _CpaCyPrimeStats Struct Reference

Collaboration diagram for _CpaCyPrimeStats:



Data Fields

- Cpa32U numPrimeTestRequests
- Cpa32U numPrimeTestRequestErrors
- Cpa32U numPrimeTestCompleted
- Cpa32U numPrimeTestCompletedErrors
- Cpa32U numPrimeTestFailures

5.42.1 Detailed Description

Prime Number Test Statistics.

Deprecated As of v1.3 of the Crypto API, this structure has been deprecated, replaced by CpaCyPrimeStats64.

Description:

This structure contains statistics on the prime number test operations. Statistics are set to zero when the component is initialized, and are collected per instance.

Definition at line 113 of file cpa_cy_prime.h.

5.42.2 Field Documentation

5.42.2.1 numPrimeTestRequests

`Cpa32U _CpaCyPrimeStats::numPrimeTestRequests`

Total number of successful prime number test requests.

Definition at line 114 of file cpa_cy_prime.h.

5.42.2.2 numPrimeTestRequestErrors

`Cpa32U _CpaCyPrimeStats::numPrimeTestRequestErrors`

Total number of prime number test requests that had an error and could not be processed.

Definition at line 116 of file cpa_cy_prime.h.

5.42.2.3 numPrimeTestCompleted

`Cpa32U _CpaCyPrimeStats::numPrimeTestCompleted`

Total number of prime number test operations that completed successfully.

Definition at line 119 of file cpa_cy_prime.h.

5.42.2.4 numPrimeTestCompletedErrors

`Cpa32U _CpaCyPrimeStats::numPrimeTestCompletedErrors`

Total number of prime number test operations that could not be completed successfully due to errors.

Definition at line 122 of file cpa_cy_prime.h.

5.42.2.5 numPrimeTestFailures

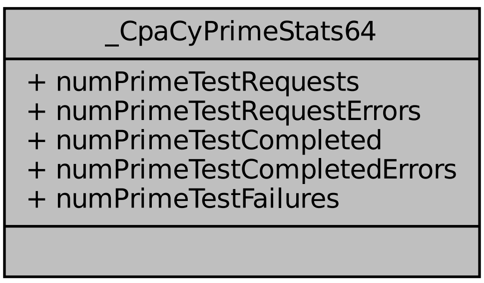
`Cpa32U _CpaCyPrimeStats::numPrimeTestFailures`

Total number of prime number test operations that executed successfully but the outcome of the test was that the number was not prime.

Definition at line 125 of file cpa_cy_prime.h.

5.43 _CpaCyPrimeStats64 Struct Reference

Collaboration diagram for _CpaCyPrimeStats64:



Data Fields

- Cpa64U numPrimeTestRequests
- Cpa64U numPrimeTestRequestErrors
- Cpa64U numPrimeTestCompleted
- Cpa64U numPrimeTestCompletedErrors
- Cpa64U numPrimeTestFailures

5.43.1 Detailed Description

Prime Number Test Statistics (64-bit version).

Description:

This structure contains a 64-bit version of the statistics on the prime number test operations. Statistics are set to zero when the component is initialized, and are collected per instance.

Definition at line 141 of file cpa_cy_prime.h.

5.43.2 Field Documentation

5.43.2.1 numPrimeTestRequests

`Cpa64U _CpaCyPrimeStats64::numPrimeTestRequests`

Total number of successful prime number test requests.

Definition at line 142 of file cpa_cy_prime.h.

5.43.2.2 numPrimeTestRequestErrors

`Cpa64U _CpaCyPrimeStats64::numPrimeTestRequestErrors`

Total number of prime number test requests that had an error and could not be processed.

Definition at line 144 of file cpa_cy_prime.h.

5.43.2.3 numPrimeTestCompleted

`Cpa64U _CpaCyPrimeStats64::numPrimeTestCompleted`

Total number of prime number test operations that completed successfully.

Definition at line 147 of file cpa_cy_prime.h.

5.43.2.4 numPrimeTestCompletedErrors

`Cpa64U _CpaCyPrimeStats64::numPrimeTestCompletedErrors`

Total number of prime number test operations that could not be completed successfully due to errors.

Definition at line 150 of file cpa_cy_prime.h.

5.43.2.5 numPrimeTestFailures

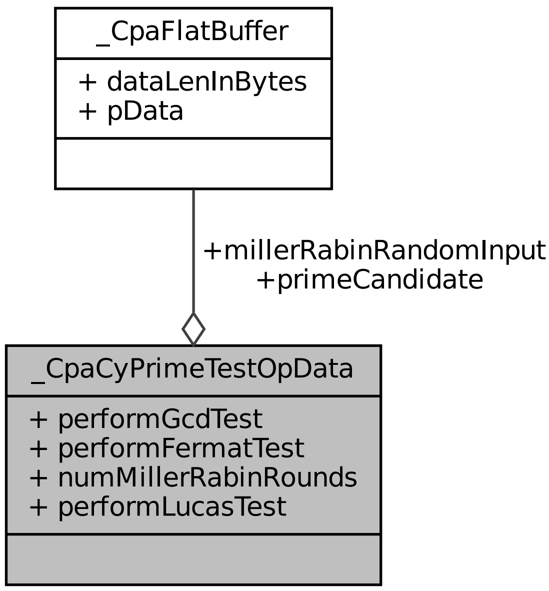
`Cpa64U _CpaCyPrimeStats64::numPrimeTestFailures`

Total number of prime number test operations that executed successfully but the outcome of the test was that the number was not prime.

Definition at line 153 of file cpa_cy_prime.h.

5.44 _CpaCyPrimeTestOpData Struct Reference

Collaboration diagram for _CpaCyPrimeTestOpData:



Data Fields

- CpaFlatBuffer primeCandidate
- CpaBoolean performGcdTest
- CpaBoolean performFermatTest
- Cpa32U numMillerRabinRounds
- CpaFlatBuffer millerRabinRandomInput
- CpaBoolean performLucasTest

5.44.1 Detailed Description

Prime Test Operation Data.

Description:

This structure contains the operation data for the cpaCyPrimeTest function. The client MUST allocate the memory for this structure and the items pointed to by this structure. When the structure is passed into the function, ownership of the memory passes to the function. Ownership of the memory returns to the client when this structure is returned in the callback function.

All values in this structure are required to be in Most Significant Byte first order, e.g. primeCandidate.pData[0] = MSB.

All numbers MUST be stored in big-endian order.

Note

If the client modifies or frees the memory referenced in this structure after it has been submitted to the cpaCyPrimeTest function, and before it has been returned in the callback, undefined behavior will result.

See also

cpaCyPrimeTest()

Definition at line 70 of file cpa_cy_prime.h.

5.44.2 Field Documentation

5.44.2.1 primeCandidate

```
CpaFlatBuffer _CpaCyPrimeTestOpData::primeCandidate
```

The prime number candidate to test

Definition at line 71 of file cpa_cy_prime.h.

5.44.2.2 performGcdTest

```
CpaBoolean _CpaCyPrimeTestOpData::performGcdTest
```

A value of CPA_TRUE means perform a GCD Primality Test

Definition at line 73 of file cpa_cy_prime.h.

5.44.2.3 performFermatTest

```
CpaBoolean _CpaCyPrimeTestOpData::performFermatTest
```

A value of CPA_TRUE means perform a Fermat Primality Test

Definition at line 75 of file cpa_cy_prime.h.

5.44.2.4 numMillerRabinRounds

```
Cpa32U _CpaCyPrimeTestOpData::numMillerRabinRounds
```

Number of Miller Rabin Primality Test rounds. Set to 0 to perform zero Miller Rabin tests. The maximum number of rounds supported is 50.

Definition at line 77 of file cpa_cy_prime.h.

5.44.2.5 millerRabinRandomInput

```
CpaFlatBuffer _CpaCyPrimeTestOpData::millerRabinRandomInput
```

Flat buffer containing a pointer to an array of n random numbers for Miller Rabin Primality Tests. The size of the buffer MUST be

```
n * (MAX(64,x))
```

where:

- n is the requested number of rounds.
- x is the minimum number of bytes required to represent the prime candidate, i.e. x = ceiling((ceiling(log2(p)))/8).

Each random number MUST be greater than 1 and less than the prime candidate - 1, with leading zeroes as necessary.

Definition at line 81 of file cpa_cy_prime.h.

5.44.2.6 performLucasTest

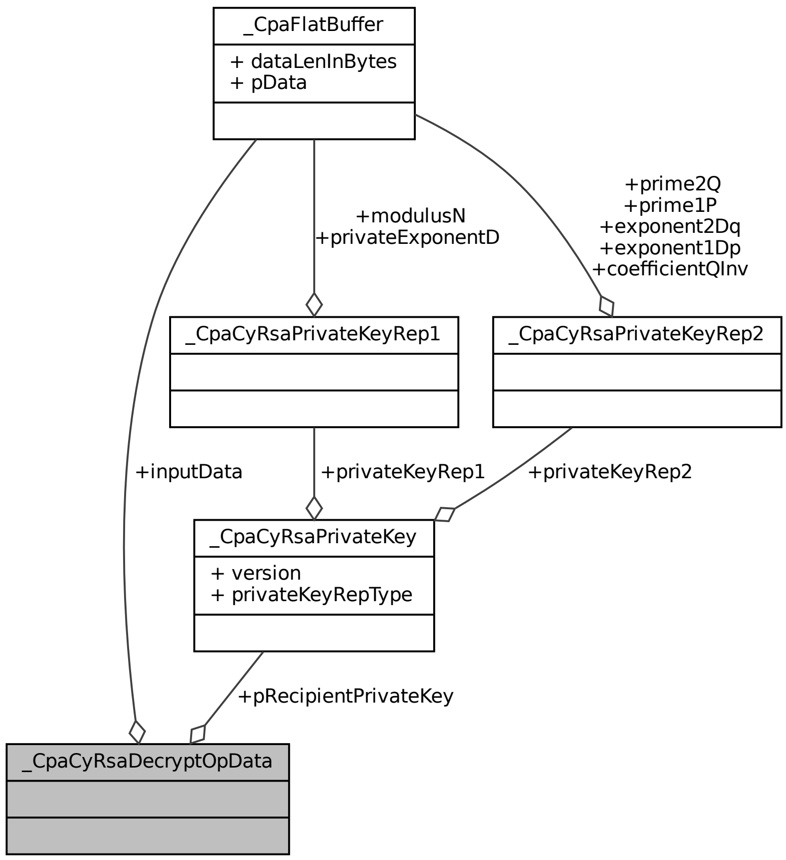
`CpaBoolean _CpaCyPrimeTestOpData::performLucasTest`

An CPA_TRUE value means perform a Lucas Primality Test

Definition at line 96 of file cpa_cy_prime.h.

5.45 _CpaCyRsaDecryptOpData Struct Reference

Collaboration diagram for _CpaCyRsaDecryptOpData:



Data Fields

- CpaCyRsaPrivateKey ∗ pRecipientPrivateKey
- CpaFlatBuffer inputData

5.45.1 Detailed Description

RSA Decryption Primitive Operation Data

Description:

This structure lists the different items that are required in the cpaCyRsaDecrypt function. As the RSA decryption primitive and signature primitive operations are mathematically identical this structure may also be used to perform an RSA signature primitive operation. When performing an RSA decryption primitive operation, the input data is the cipher text and the output data is the message text. When performing an RSA signature primitive operation, the input data is the message and the output data is the signature. The client MUST allocate the memory for this structure. When the structure is passed into the function, ownership of the memory passes to he function. Ownership of the memory returns to the client when this structure is returned in the CpaCyRsaDecryptCbFunc callback function.

Note

If the client modifies or frees the memory referenced in this structure after it has been submitted to the cpaCyRsaDecrypt function, and before it has been returned in the callback, undefined behavior will result. All values in this structure are required to be in Most Significant Byte first order, e.g. inputData.pData[0] = MSB.

Definition at line 340 of file cpa_cy_rsa.h.

5.45.2 Field Documentation

5.45.2.1 pRecipientPrivateKey

`CpaCyRsaPrivateKey* _CpaCyRsaDecryptOpData::pRecipientPrivateKey`

Pointer to the recipient's RSA private key.

Definition at line 341 of file cpa_cy_rsa.h.

5.45.2.2 inputData

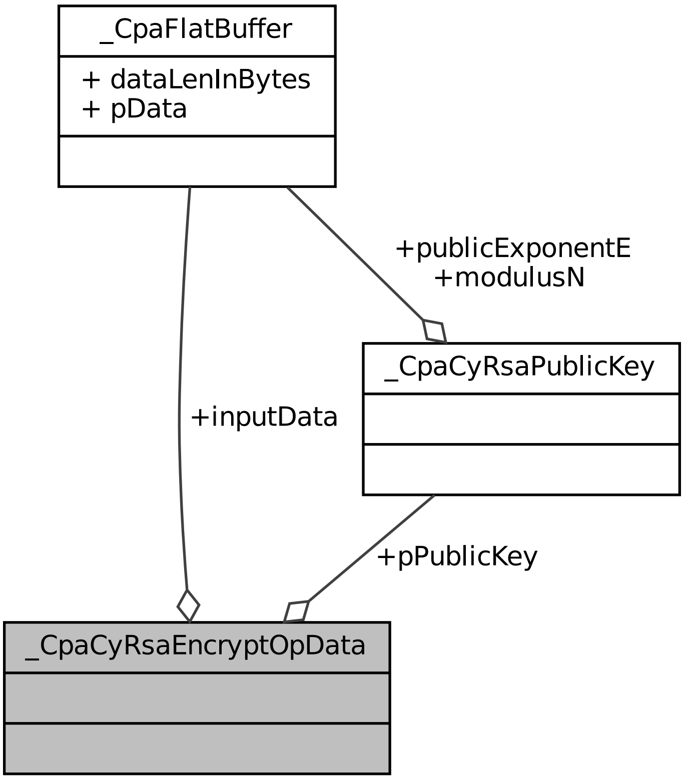
`CpaFlatBuffer _CpaCyRsaDecryptOpData::inputData`

The input data that the RSA decryption primitive operation is performed on. The data pointed to is an integer that MUST be in big- endian order. The value MUST be between 0 and the modulus n - 1.

Definition at line 343 of file cpa_cy_rsa.h.

5.46 _CpaCyRsaEncryptOpData Struct Reference

Collaboration diagram for _CpaCyRsaEncryptOpData:



Data Fields

- CpaCyRsaPublicKey ∗ pPublicKey
- CpaFlatBuffer inputData

5.46.1 Detailed Description

RSA Encryption Primitive Operation Data

Description:

This structure lists the different items that are required in the cpaCyRsaEncrypt function. As the RSA encryption primitive and verification primitive operations are mathematically identical this structure may also be used to perform an RSA verification primitive operation. When performing an RSA encryption primitive operation, the input data is the message and the output data is the cipher text. When performing an RSA verification primitive operation, the input data is the signature and the output data is the message. The client MUST allocate the memory for this structure. When the structure is passed into the function, ownership of the memory passes to the function. Ownership of the memory returns to the client when this structure is returned in the CpaCyRsaEncryptCbFunc callback function.

Note

If the client modifies or frees the memory referenced in this structure after it has been submitted to the cpaCyRsaEncrypt function, and before it has been returned in the callback, undefined behavior will result. All values in this structure are required to be in Most Significant Byte first order, e.g. inputData.pData[0] = MSB.

Definition at line 302 of file cpa_cy_rsa.h.

5.46.2 Field Documentation

5.46.2.1 pPublicKey

`CpaCyRsaPublicKey* _CpaCyRsaEncryptOpData::pPublicKey`

Pointer to the public key.

Definition at line 303 of file cpa_cy_rsa.h.

5.46.2.2 inputData

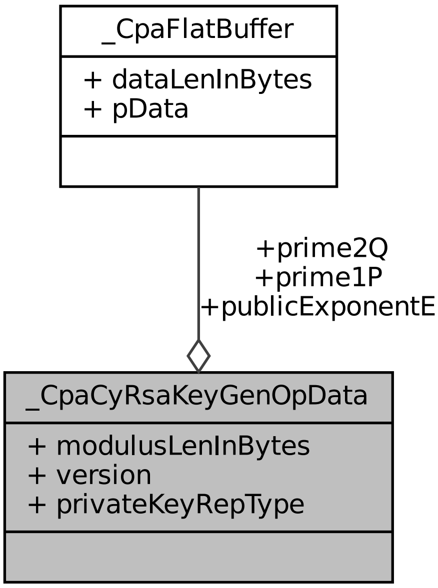
`CpaFlatBuffer _CpaCyRsaEncryptOpData::inputData`

The input data that the RSA encryption primitive operation is performed on. The data pointed to is an integer that MUST be in big- endian order. The value MUST be between 0 and the modulus n - 1.

Definition at line 305 of file cpa_cy_rsa.h.

5.47 _CpaCyRsaKeyGenOpData Struct Reference

Collaboration diagram for _CpaCyRsaKeyGenOpData:



Data Fields

- CpaFlatBuffer prime1P
- CpaFlatBuffer prime2Q
- Cpa32U modulusLenInBytes
- CpaCyRsaVersion version
- CpaCyRsaPrivateKeyRepType privateKeyRepType
- CpaFlatBuffer publicExponentE

5.47.1 Detailed Description

RSA Key Generation Data.

Description:

This structure lists the different items that are required in the cpaCyRsaGenKey function. The client MUST allocate the memory for this structure. When the structure is passed into the function, ownership of the memory passes to the function. Ownership of the memory returns to the client when this structure is returned in the CpaCyRsaKeyGenCbFunc callback function.

Note

If the client modifies or frees the memory referenced in this structure after it has been submitted to the cpaCyRsaGenKey function, and before it has been returned in the callback, undefined behavior will result. All values in this structure are required to be in Most Significant Byte first order, e.g. prime1P.pData[0] = MSB.

The following limitations on the permutations of the supported bit lengths of p, q and n (written as {p, q, n}) apply:

- {256, 256, 512} or
- {512, 512, 1024} or
- {768, 768, 1536} or
- {1024, 1024, 2048} or
- {1536, 1536, 3072} or
- {2048, 2048, 4096}.

Definition at line 237 of file cpa_cy_rsa.h.

5.47.2 Field Documentation

5.47.2.1 prime1P

`CpaFlatBuffer _CpaCyRsaKeyGenOpData::prime1P`

A large random prime number (p). This MUST be created by the client. Permitted bit lengths are: 256, 512, 768, 1024, 1536 or 2048. Limitations apply - refer to the description above for details.

Definition at line 238 of file cpa_cy_rsa.h.

5.47.2.2 prime2Q

`CpaFlatBuffer _CpaCyRsaKeyGenOpData::prime2Q`

A large random prime number (q). This MUST be created by the client. Permitted bit lengths are: 256, 512, 768, 1024, 1536 or 2048. Limitations apply - refer to the description above for details. If the private key representation type is 2, then this pointer will be assigned to the relevant structure member of the representation 2 private key.

Definition at line 242 of file cpa_cy_rsa.h.

5.47.2.3 modulusLenInBytes

`Cpa32U _CpaCyRsaKeyGenOpData::modulusLenInBytes`

The bit length of the modulus (n). This is the modulus length for both the private and public keys. The length of the modulus N parameter for the private key representation 1 structure and the public key structures will be assigned to this value. References to the strength of RSA actually refer to this bit length. Recommended minimum is 1024 bits. Permitted lengths are:

- 512 bits (64 bytes),
- 1024 bits (128 bytes),
- 1536 bits (192 bytes),
- 2048 bits (256 bytes),
- 3072 bits (384 bytes), or
- 4096 bits (512 bytes). Limitations apply - refer to description above for details.

Definition at line 248 of file cpa_cy_rsa.h.

5.47.2.4 version

`CpaCyRsaVersion _CpaCyRsaKeyGenOpData::version`

Indicates the version of the PKCS #1 specification that is supported. Note that this applies to both representations.

Definition at line 262 of file cpa_cy_rsa.h.

5.47.2.5 privateKeyRepType

`CpaCyRsaPrivateKeyRepType _CpaCyRsaKeyGenOpData::privateKeyRepType`

This value is used to identify which of the private key representation types is required to be generated.

Definition at line 266 of file cpa_cy_rsa.h.

5.47.2.6 publicExponentE

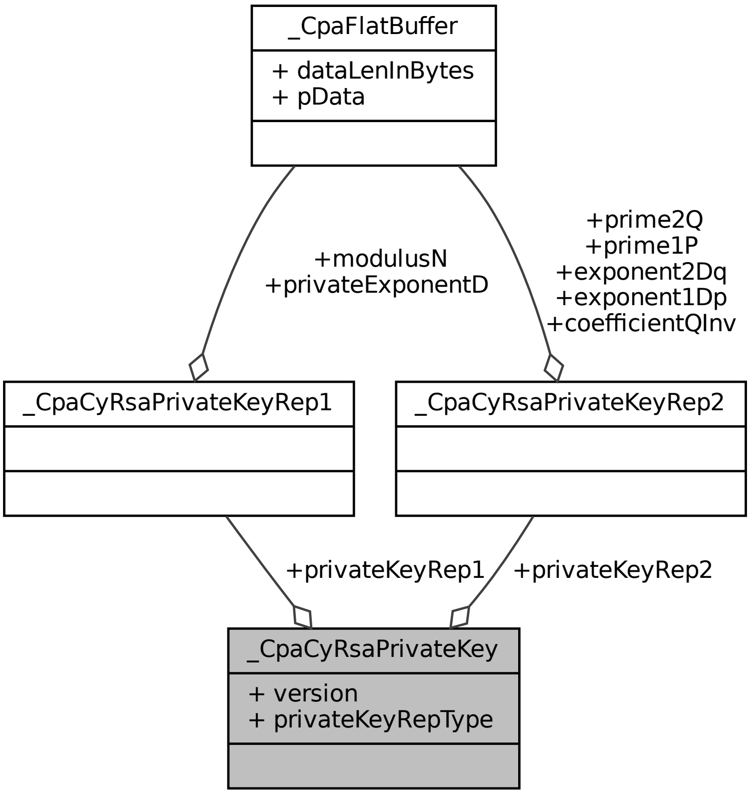
`CpaFlatBuffer _CpaCyRsaKeyGenOpData::publicExponentE`

The public exponent (e).

Definition at line 269 of file cpa_cy_rsa.h.

5.48 _CpaCyRsaPrivateKey Struct Reference

Collaboration diagram for _CpaCyRsaPrivateKey:



Data Fields

- CpaCyRsaVersion version
- CpaCyRsaPrivateKeyRepType privateKeyRepType
- CpaCyRsaPrivateKeyRep1 privateKeyRep1
- CpaCyRsaPrivateKeyRep2 privateKeyRep2

5.48.1 Detailed Description

RSA Private Key Structure.

Description:

This structure contains the two representations that can be used for describing the RSA private key. The privateKeyRepType will be used to identify which representation is to be used. Typically, using the second representation results in faster decryption operations.

Definition at line 181 of file cpa_cy_rsa.h.

5.48.2 Field Documentation

5.48.2.1 version

`CpaCyRsaVersion _CpaCyRsaPrivateKey::version`

Indicates the version of the PKCS #1 specification that is supported. Note that this applies to both representations.

Definition at line 182 of file cpa_cy_rsa.h.

5.48.2.2 privateKeyRepType

`CpaCyRsaPrivateKeyRepType _CpaCyRsaPrivateKey::privateKeyRepType`

This value is used to identify which of the private key representation types in this structure is relevant. When performing key generation operations for Type 2 representations, memory must also be allocated for the type 1 representations, and values for both will be returned.

Definition at line 186 of file cpa_cy_rsa.h.

5.48.2.3 privateKeyRep1

`CpaCyRsaPrivateKeyRep1 _CpaCyRsaPrivateKey::privateKeyRep1`

This is the first representation of the RSA private key as defined in the PKCS #1 V2.1 specification. For key generation operations the memory for this structure is allocated by the client and the specific values are generated. For other operations this is an input parameter.

Definition at line 192 of file cpa_cy_rsa.h.

5.48.2.4 privateKeyRep2

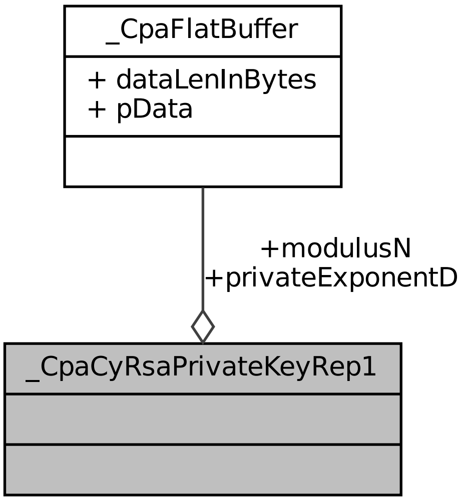
`CpaCyRsaPrivateKeyRep2 _CpaCyRsaPrivateKey::privateKeyRep2`

This is the second representation of the RSA private key as defined in the PKCS #1 V2.1 specification. For key generation operations the memory for this structure is allocated by the client and the specific values are generated. For other operations this is an input parameter.

Definition at line 198 of file cpa_cy_rsa.h.

5.49 _CpaCyRsaPrivateKeyRep1 Struct Reference

Collaboration diagram for _CpaCyRsaPrivateKeyRep1:



Data Fields

- CpaFlatBuffer modulusN
- CpaFlatBuffer privateExponentD

5.49.1 Detailed Description

RSA Private Key Structure For Representation 1.

Description:

This structure contains the first representation that can be used for describing the RSA private key, represented by the tuple of the modulus (n) and the private exponent (d). All values in this structure are required to be in Most Significant Byte first order, e.g. modulusN.pData[0] = MSB.

Definition at line 101 of file cpa_cy_rsa.h.

5.49.2 Field Documentation

5.49.2.1 modulusN

CpaFlatBuffer _CpaCyRsaPrivateKeyRep1::modulusN

The modulus (n). For key generation operations the memory MUST be allocated by the client and the value is generated. For other operations this is an input. Permitted lengths are:

- 512 bits (64 bytes),
- 1024 bits (128 bytes),
- 1536 bits (192 bytes),
- 2048 bits (256 bytes),
- 3072 bits (384 bytes),
- 4096 bits (512 bytes), or
- 8192 bits (1024 bytes).

Definition at line 102 of file cpa_cy_rsa.h.

5.49.2.2 privateExponentD

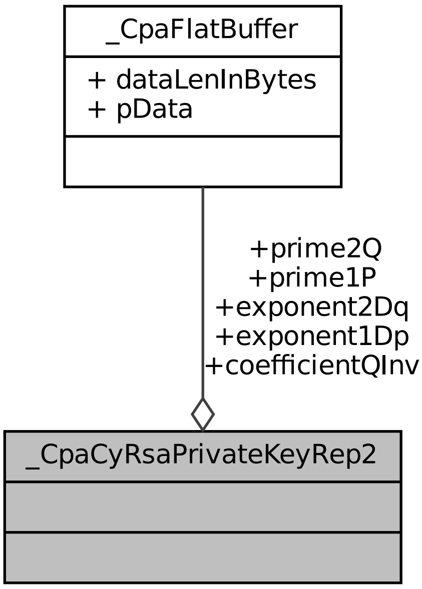
CpaFlatBuffer _CpaCyRsaPrivateKeyRep1::privateExponentD

The private exponent (d). For key generation operations the memory MUST be allocated by the client and the value is generated. For other operations this is an input. NOTE: It is important that the value D is big enough. It is STRONGLY recommended that this value is at least half the length of the modulus N to protect against the Wiener attack.

Definition at line 115 of file cpa_cy_rsa.h.

5.50 _CpaCyRsaPrivateKeyRep2 Struct Reference

Collaboration diagram for _CpaCyRsaPrivateKeyRep2:



Data Fields

- CpaFlatBuffer prime1P
- CpaFlatBuffer prime2Q
- CpaFlatBuffer exponent1Dp
- CpaFlatBuffer exponent2Dq
- CpaFlatBuffer coefficientQInv

5.50.1 Detailed Description

RSA Private Key Structure For Representation 2.

Description:

This structure contains the second representation that can be used for describing the RSA private key. The quintuple of p, q, dP, dQ, and qInv (explained below and in the spec) are required for the second representation. The optional sequence of triplets are not included. All values in this structure are required to be in Most Significant Byte first order, e.g. prime1P.pData[0] = MSB.

Definition at line 137 of file cpa_cy_rsa.h.

5.50.2 Field Documentation

5.50.2.1 prime1P

`CpaFlatBuffer _CpaCyRsaPrivateKeyRep2::prime1P`

The first large prime (p). For key generation operations, this field is unused.

Definition at line 138 of file cpa_cy_rsa.h.

5.50.2.2 prime2Q

`CpaFlatBuffer _CpaCyRsaPrivateKeyRep2::prime2Q`

The second large prime (q). For key generation operations, this field is unused.

Definition at line 141 of file cpa_cy_rsa.h.

5.50.2.3 exponent1Dp

`CpaFlatBuffer _CpaCyRsaPrivateKeyRep2::exponent1Dp`

The first factor CRT exponent (dP). d mod (p-1).

Definition at line 144 of file cpa_cy_rsa.h.

5.50.2.4 exponent2Dq

`CpaFlatBuffer _CpaCyRsaPrivateKeyRep2::exponent2Dq`

The second factor CRT exponent (dQ). d mod (q-1).

Definition at line 146 of file cpa_cy_rsa.h.

5.50.2.5 coefficientQInv

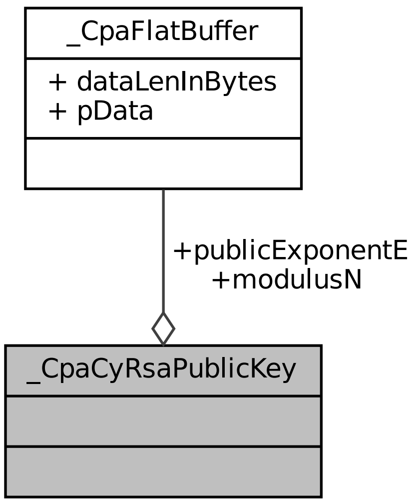
`CpaFlatBuffer _CpaCyRsaPrivateKeyRep2::coefficientQInv`

The (first) Chinese Remainder Theorem (CRT) coefficient (qInv). (inverse of q) mod p.

Definition at line 148 of file cpa_cy_rsa.h.

5.51 _CpaCyRsaPublicKey Struct Reference

Collaboration diagram for _CpaCyRsaPublicKey:



Data Fields

- CpaFlatBuffer modulusN
- CpaFlatBuffer publicExponentE

5.51.1 Detailed Description

RSA Public Key Structure.

Description:

This structure contains the two components which comprise the RSA public key as defined in the PKCS #1 V2.1 standard. All values in this structure are required to be in Most Significant Byte first order, e.g. modulusN.pData[0] = MSB.

Definition at line 73 of file cpa_cy_rsa.h.

5.51.2 Field Documentation

5.51.2.1 modulusN

`CpaFlatBuffer _CpaCyRsaPublicKey::modulusN`

The modulus (n). For key generation operations, the client MUST allocate the memory for this parameter; its value is generated. For encrypt operations this parameter is an input.

Definition at line 74 of file cpa_cy_rsa.h.

5.51.2.2 publicExponentE

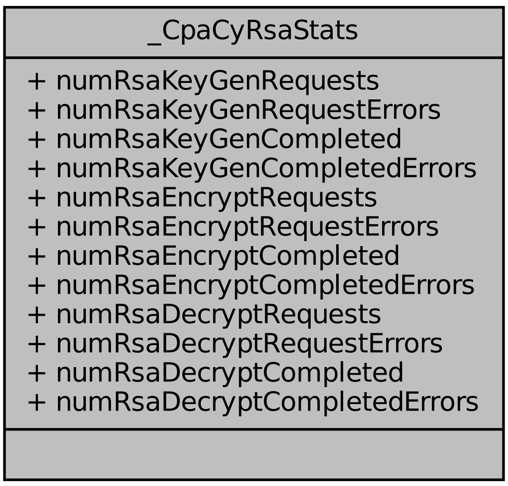
`CpaFlatBuffer _CpaCyRsaPublicKey::publicExponentE`

The public exponent (e). For key generation operations, this field is unused. It is NOT generated by the interface; it is the responsibility of the client to set this to the same value as the corresponding parameter on the CpaCyRsaKeyGenOpData structure before using the key for encryption. For encrypt operations this parameter is an input.

Definition at line 79 of file cpa_cy_rsa.h.

5.52 _CpaCyRsaStats Struct Reference

Collaboration diagram for _CpaCyRsaStats:



Data Fields

- Cpa32U numRsaKeyGenRequests
- Cpa32U numRsaKeyGenRequestErrors
- Cpa32U numRsaKeyGenCompleted
- Cpa32U numRsaKeyGenCompletedErrors
- Cpa32U numRsaEncryptRequests
- Cpa32U numRsaEncryptRequestErrors
- Cpa32U numRsaEncryptCompleted
- Cpa32U numRsaEncryptCompletedErrors
- Cpa32U numRsaDecryptRequests
- Cpa32U numRsaDecryptRequestErrors
- Cpa32U numRsaDecryptCompleted
- Cpa32U numRsaDecryptCompletedErrors

5.52.1 Detailed Description

RSA Statistics.

Deprecated As of v1.3 of the Crypto API, this structure has been deprecated, replaced by CpaCyRsaStats64.

Description:

This structure contains statistics on the RSA operations. Statistics are set to zero when the component is initialized, and are collected per instance.

Definition at line 361 of file cpa_cy_rsa.h.

5.52.2 Field Documentation

5.52.2.1 numRsaKeyGenRequests

`Cpa32U _CpaCyRsaStats::numRsaKeyGenRequests`

Total number of successful RSA key generation requests.

Definition at line 362 of file cpa_cy_rsa.h.

5.52.2.2 numRsaKeyGenRequestErrors

`Cpa32U _CpaCyRsaStats::numRsaKeyGenRequestErrors`

Total number of RSA key generation requests that had an error and could not be processed.

Definition at line 364 of file cpa_cy_rsa.h.

5.52.2.3 numRsaKeyGenCompleted

`Cpa32U _CpaCyRsaStats::numRsaKeyGenCompleted`

Total number of RSA key generation operations that completed successfully.

Definition at line 367 of file cpa_cy_rsa.h.

5.52.2.4 numRsaKeyGenCompletedErrors

`Cpa32U _CpaCyRsaStats::numRsaKeyGenCompletedErrors`

Total number of RSA key generation operations that could not be completed successfully due to errors.

Definition at line 370 of file cpa_cy_rsa.h.

5.52.2.5 numRsaEncryptRequests

`Cpa32U _CpaCyRsaStats::numRsaEncryptRequests`

Total number of successful RSA encrypt operation requests.

Definition at line 373 of file cpa_cy_rsa.h.

5.52.2.6 numRsaEncryptRequestErrors

`Cpa32U _CpaCyRsaStats::numRsaEncryptRequestErrors`

Total number of RSA encrypt requests that had an error and could not be processed.

Definition at line 375 of file cpa_cy_rsa.h.

5.52.2.7 numRsaEncryptCompleted

`Cpa32U _CpaCyRsaStats::numRsaEncryptCompleted`

Total number of RSA encrypt operations that completed successfully.

Definition at line 378 of file cpa_cy_rsa.h.

5.52.2.8 numRsaEncryptCompletedErrors

`Cpa32U _CpaCyRsaStats::numRsaEncryptCompletedErrors`

Total number of RSA encrypt operations that could not be completed successfully due to errors.

Definition at line 381 of file cpa_cy_rsa.h.

5.52.2.9 numRsaDecryptRequests

`Cpa32U _CpaCyRsaStats::numRsaDecryptRequests`

Total number of successful RSA decrypt operation requests.

Definition at line 384 of file cpa_cy_rsa.h.

5.52.2.10 numRsaDecryptRequestErrors

`Cpa32U _CpaCyRsaStats::numRsaDecryptRequestErrors`

Total number of RSA decrypt requests that had an error and could not be processed.

Definition at line 386 of file cpa_cy_rsa.h.

5.52.2.11 numRsaDecryptCompleted

`Cpa32U _CpaCyRsaStats::numRsaDecryptCompleted`

Total number of RSA decrypt operations that completed successfully.

Definition at line 389 of file cpa_cy_rsa.h.

5.52.2.12 numRsaDecryptCompletedErrors

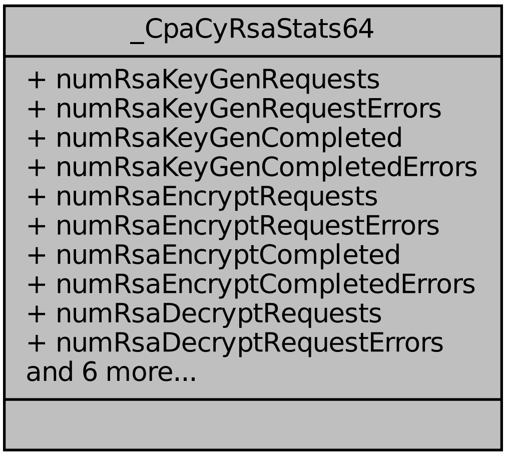
`Cpa32U _CpaCyRsaStats::numRsaDecryptCompletedErrors`

Total number of RSA decrypt operations that could not be completed successfully due to errors.

Definition at line 392 of file cpa_cy_rsa.h.

5.53 _CpaCyRsaStats64 Struct Reference

Collaboration diagram for _CpaCyRsaStats64:



Data Fields

- Cpa64U numRsaKeyGenRequests
- Cpa64U numRsaKeyGenRequestErrors
- Cpa64U numRsaKeyGenCompleted
- Cpa64U numRsaKeyGenCompletedErrors
- Cpa64U numRsaEncryptRequests
- Cpa64U numRsaEncryptRequestErrors
- Cpa64U numRsaEncryptCompleted
- Cpa64U numRsaEncryptCompletedErrors
- Cpa64U numRsaDecryptRequests
- Cpa64U numRsaDecryptRequestErrors
- Cpa64U numRsaDecryptCompleted
- Cpa64U numRsaDecryptCompletedErrors
- Cpa64U numKptRsaDecryptRequests
- Cpa64U numKptRsaDecryptRequestErrors
- Cpa64U numKptRsaDecryptCompleted
- Cpa64U numKptRsaDecryptCompletedErrors

5.53.1 Detailed Description

RSA Statistics (64-bit version).

Description:

This structure contains 64-bit version of the statistics on the RSA operations. Statistics are set to zero when the component is initialized, and are collected per instance.

Definition at line 407 of file cpa_cy_rsa.h.

5.53.2 Field Documentation

5.53.2.1 numRsaKeyGenRequests

`Cpa64U _CpaCyRsaStats64::numRsaKeyGenRequests`

Total number of successful RSA key generation requests.

Definition at line 408 of file cpa_cy_rsa.h.

5.53.2.2 numRsaKeyGenRequestErrors

`Cpa64U _CpaCyRsaStats64::numRsaKeyGenRequestErrors`

Total number of RSA key generation requests that had an error and could not be processed.

Definition at line 410 of file cpa_cy_rsa.h.

5.53.2.3 numRsaKeyGenCompleted

`Cpa64U _CpaCyRsaStats64::numRsaKeyGenCompleted`

Total number of RSA key generation operations that completed successfully.

Definition at line 413 of file cpa_cy_rsa.h.

5.53.2.4 numRsaKeyGenCompletedErrors

`Cpa64U _CpaCyRsaStats64::numRsaKeyGenCompletedErrors`

Total number of RSA key generation operations that could not be completed successfully due to errors.

Definition at line 416 of file cpa_cy_rsa.h.

5.53.2.5 numRsaEncryptRequests

`Cpa64U _CpaCyRsaStats64::numRsaEncryptRequests`

Total number of successful RSA encrypt operation requests.

Definition at line 419 of file cpa_cy_rsa.h.

5.53.2.6 numRsaEncryptRequestErrors

`Cpa64U _CpaCyRsaStats64::numRsaEncryptRequestErrors`

Total number of RSA encrypt requests that had an error and could not be processed.

Definition at line 421 of file cpa_cy_rsa.h.

5.53.2.7 numRsaEncryptCompleted

`Cpa64U _CpaCyRsaStats64::numRsaEncryptCompleted`

Total number of RSA encrypt operations that completed successfully.

Definition at line 424 of file cpa_cy_rsa.h.

5.53.2.8 numRsaEncryptCompletedErrors

`Cpa64U _CpaCyRsaStats64::numRsaEncryptCompletedErrors`

Total number of RSA encrypt operations that could not be completed successfully due to errors.

Definition at line 427 of file cpa_cy_rsa.h.

5.53.2.9 numRsaDecryptRequests

`Cpa64U _CpaCyRsaStats64::numRsaDecryptRequests`

Total number of successful RSA decrypt operation requests.

Definition at line 430 of file cpa_cy_rsa.h.

5.53.2.10 numRsaDecryptRequestErrors

`Cpa64U _CpaCyRsaStats64::numRsaDecryptRequestErrors`

Total number of RSA decrypt requests that had an error and could not be processed.

Definition at line 432 of file cpa_cy_rsa.h.

5.53.2.11 numRsaDecryptCompleted

`Cpa64U _CpaCyRsaStats64::numRsaDecryptCompleted`

Total number of RSA decrypt operations that completed successfully.

Definition at line 435 of file cpa_cy_rsa.h.

5.53.2.12 numRsaDecryptCompletedErrors

`Cpa64U _CpaCyRsaStats64::numRsaDecryptCompletedErrors`

Total number of RSA decrypt operations that could not be completed successfully due to errors.

Definition at line 438 of file cpa_cy_rsa.h.

5.53.2.13 numKptRsaDecryptRequests

`Cpa64U _CpaCyRsaStats64::numKptRsaDecryptRequests`

Total number of successful KPT RSA decrypt operation requests.

Definition at line 441 of file cpa_cy_rsa.h.

5.53.2.14 numKptRsaDecryptRequestErrors

`Cpa64U _CpaCyRsaStats64::numKptRsaDecryptRequestErrors`

Total number of KPT RSA decrypt requests that had an error and could not be processed.

Definition at line 443 of file cpa_cy_rsa.h.

5.53.2.15 numKptRsaDecryptCompleted

`Cpa64U _CpaCyRsaStats64::numKptRsaDecryptCompleted`

Total number of KPT RSA decrypt operations that completed successfully.

Definition at line 446 of file cpa_cy_rsa.h.

5.53.2.16 numKptRsaDecryptCompletedErrors

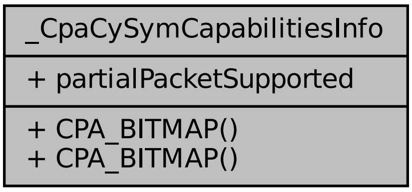
`Cpa64U _CpaCyRsaStats64::numKptRsaDecryptCompletedErrors`

Total number of KPT RSA decrypt operations that could not be completed successfully due to errors.

Definition at line 449 of file cpa_cy_rsa.h.

5.54 _CpaCySymCapabilitiesInfo Struct Reference

Collaboration diagram for _CpaCySymCapabilitiesInfo:



Public Member Functions

- CPA_BITMAP (ciphers, CPA_CY_SYM_CIPHER_CAP_BITMAP_SIZE)
- CPA_BITMAP (hashes, CPA_CY_SYM_HASH_CAP_BITMAP_SIZE)

Data Fields

- CpaBoolean partialPacketSupported

5.54.1 Detailed Description

Symmetric Capabilities Info

Description:

This structure contains the capabilities that vary across implementations of the symmetric sub-API of the cryptographic API. This structure is used in conjunction with cpaCySymQueryCapabilities() to determine the capabilities supported by a particular API implementation.

For example, to see if an implementation supports cipher CPA_CY_SYM_CIPHER_AES_CBC, use the code

```
if (CPA_BITMAP_BIT_TEST(capInfo.ciphers, CPA_CY_SYM_CIPHER_AES_CBC))
{
    // algo is supported
}
else
{
    // algo is not supported
}
```

```
 The client MUST allocate memory for this structure and any members
 that require memory.  When the structure is passed into the function
 ownership of the memory passes to the function. Ownership of the
 memory returns to the client when the function returns.
```

Definition at line 1751 of file cpa_cy_sym.h.

5.54.2 Member Function Documentation

5.54.2.1 CPA_BITMAP() [1/2]

```
_CpaCySymCapabilitiesInfo::CPA_BITMAP (
            ciphers ,
            CPA_CY_SYM_CIPHER_CAP_BITMAP_SIZE )
```

Bitmap representing which cipher algorithms (and modes) are supported by the instance. Bits can be tested using the macro CPA_BITMAP_BIT_TEST. The bit positions are those specified in the enumerated type CpaCySymCipherAlgorithm.

5.54.2.2 CPA_BITMAP() [2/2]

```
_CpaCySymCapabilitiesInfo::CPA_BITMAP (
            hashes ,
            CPA_CY_SYM_HASH_CAP_BITMAP_SIZE )
```

Bitmap representing which hash/authentication algorithms are supported by the instance. Bits can be tested using the macro CPA_BITMAP_BIT_TEST. The bit positions are those specified in the enumerated type CpaCySymHashAlgorithm.

5.54.3 Field Documentation

5.54.3.1 partialPacketSupported

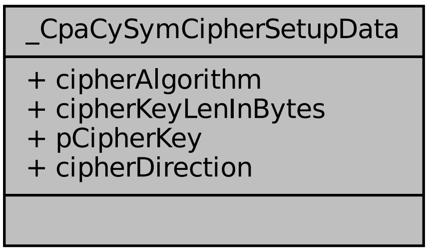
```
CpaBoolean _CpaCySymCapabilitiesInfo::partialPacketSupported
```

CPA_TRUE if instance supports partial packets. See CpaCySymPacketType.

Definition at line 1765 of file cpa_cy_sym.h.

5.55 _CpaCySymCipherSetupData Struct Reference

Collaboration diagram for _CpaCySymCipherSetupData:



Data Fields

- CpaCySymCipherAlgorithm cipherAlgorithm
- Cpa32U cipherKeyLenInBytes
- Cpa8U * pCipherKey
- CpaCySymCipherDirection cipherDirection

5.55.1 Detailed Description

Symmetric Cipher Setup Data.

Description:

This structure contains data relating to Cipher (Encryption and Decryption) to set up a session.

Definition at line 229 of file cpa_cy_sym.h.

5.55.2 Field Documentation

5.55.2.1 cipherAlgorithm

`CpaCySymCipherAlgorithm _CpaCySymCipherSetupData::cipherAlgorithm`

Cipher algorithm and mode

Definition at line 230 of file cpa_cy_sym.h.

5.55.2.2 cipherKeyLenInBytes

`Cpa32U _CpaCySymCipherSetupData::cipherKeyLenInBytes`

Cipher key length in bytes. For AES it can be 128 bits (16 bytes), 192 bits (24 bytes) or 256 bits (32 bytes). For the CCM mode of operation, the only supported key length is 128 bits (16 bytes). For the CPA_CY_SYM_CIPHER_AES_F8 mode of operation, cipherKeyLenInBytes should be set to the combined length of the encryption key and the keymask. Since the keymask and the encryption key are the same size, cipherKeyLenInBytes should be set to 2 x the AES encryption key length. For the AES-XTS mode of operation:

- Two keys must be provided and cipherKeyLenInBytes refers to total length of the two keys.
- Each key can be either 128 bits (16 bytes) or 256 bits (32 bytes).
- Both keys must have the same size.

Definition at line 232 of file cpa_cy_sym.h.

5.55.2.3 pCipherKey

```
Cpa8U* _CpaCySymCipherSetupData::pCipherKey
```

Cipher key For the CPA_CY_SYM_CIPHER_AES_F8 mode of operation, pCipherKey will point to a concatenation of the AES encryption key followed by a keymask. As per RFC3711, the keymask should be padded with trailing bytes to match the length of the encryption key used. For AES-XTS mode of operation, two keys must be provided and pCipherKey must point to the two keys concatenated together (Key1 || Key2). cipherKeyLenInBytes will contain the total size of both keys.

Definition at line 246 of file cpa_cy_sym.h.

5.55.2.4 cipherDirection

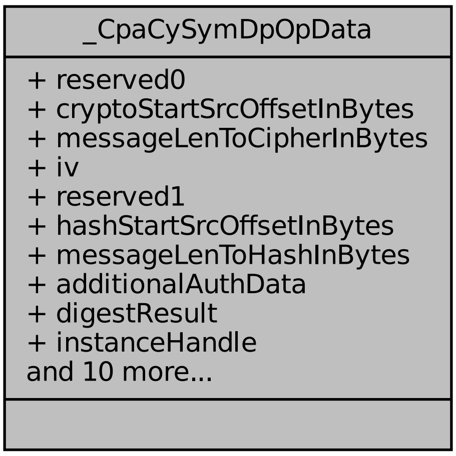
```
CpaCySymCipherDirection _CpaCySymCipherSetupData::cipherDirection
```

This parameter determines if the cipher operation is an encrypt or a decrypt operation. For the RC4 algorithm and the F8/CTR modes, only encrypt operations are valid.

Definition at line 255 of file cpa_cy_sym.h.

5.56 _CpaCySymDpOpData Struct Reference

Collaboration diagram for _CpaCySymDpOpData:



Data Fields

- Cpa64U reserved0
- Cpa32U cryptoStartSrcOffsetInBytes
- Cpa32U messageLenToCipherInBytes
- CpaPhysicalAddr iv
- Cpa64U reserved1
- Cpa32U hashStartSrcOffsetInBytes
- Cpa32U messageLenToHashInBytes
- CpaPhysicalAddr additionalAuthData
- CpaPhysicalAddr digestResult
- CpaInstanceHandle instanceHandle
- CpaCySymDpSessionCtx sessionCtx
- Cpa32U ivLenInBytes
- CpaPhysicalAddr srcBuffer
- Cpa32U srcBufferLen
- CpaPhysicalAddr dstBuffer
- Cpa32U dstBufferLen
- CpaPhysicalAddr thisPhys
- Cpa8U ∗ pIv
- Cpa8U ∗ pAdditionalAuthData
- void ∗ pCallbackTag

5.56.1 Detailed Description

Operation Data for cryptographic data plane API.

Description:

This structure contains data relating to a request to perform symmetric cryptographic processing on one or more data buffers.

The physical memory to which this structure points needs to be at least 8-byte aligned.

All reserved fields SHOULD NOT be written or read by the calling code.

See also

cpaCySymDpEnqueueOp, cpaCySymDpEnqueueOpBatch

Definition at line 132 of file cpa_cy_sym_dp.h.

5.56.2 Field Documentation

5.56.2.1 reserved0

`Cpa64U _CpaCySymDpOpData::reserved0`

Reserved for internal usage.

Definition at line 133 of file cpa_cy_sym_dp.h.

5.56.2.2 cryptoStartSrcOffsetInBytes

`Cpa32U _CpaCySymDpOpData::cryptoStartSrcOffsetInBytes`

Starting point for cipher processing, specified as number of bytes from start of data in the source buffer. The result of the cipher operation will be written back into the buffer starting at this location in the destination buffer.

Definition at line 135 of file cpa_cy_sym_dp.h.

5.56.2.3 messageLenToCipherInBytes

`Cpa32U _CpaCySymDpOpData::messageLenToCipherInBytes`

The message length, in bytes, of the source buffer on which the cryptographic operation will be computed. This must be a multiple of the block size if a block cipher is being used. This is also the same as the result length.

Note

> In the case of CCM (CPA_CY_SYM_HASH_AES_CCM), this value should not include the length of the padding or the length of the MAC; the driver will compute the actual number of bytes over which the encryption will occur, which will include these values.
>
> For AES-GMAC (CPA_CY_SYM_HASH_AES_GMAC), this field should be set to 0.
>
> On some implementations, this length may be limited to a 16-bit value (65535 bytes).

Definition at line 141 of file cpa_cy_sym_dp.h.

5.56.2.4 iv

`CpaPhysicalAddr _CpaCySymDpOpData::iv`

Initialization Vector or Counter. Specifically, this is the physical address of one of the following:

- For block ciphers in CBC mode, or for Kasumi in F8 mode, or for SNOW3G in UEA2 mode, this is the Initialization Vector (IV) value.
- For ARC4, this is reserved for internal usage.
- For block ciphers in CTR mode, this is the counter.
- For GCM mode, this is either the IV (if the length is 96 bits) or J0 (for other sizes), where J0 is as defined by NIST SP800-38D. Regardless of the IV length, a full 16 bytes needs to be allocated.
- For CCM mode, the first byte is reserved, and the nonce should be written starting at &pIv[1] (to allow space for the implementation to write in the flags in the first byte). Note that a full 16 bytes should be allocated, even though the ivLenInBytes field will have a value less than this. The macro CPA_CY_SYM_CCM_SET_NONCE may be used here.

Definition at line 158 of file cpa_cy_sym_dp.h.

5.56.2.5 reserved1

`Cpa64U _CpaCySymDpOpData::reserved1`

Reserved for internal usage.

Definition at line 177 of file cpa_cy_sym_dp.h.

5.56.2.6 hashStartSrcOffsetInBytes

`Cpa32U _CpaCySymDpOpData::hashStartSrcOffsetInBytes`

Starting point for hash processing, specified as number of bytes from start of packet in source buffer.

Note

For CCM and GCM modes of operation, this value in this field is ignored, and the field is reserved for internal usage. The fields additionalAuthData and pAdditionalAuthData should be set instead.

For AES-GMAC (CPA_CY_SYM_HASH_AES_GMAC) mode of operation, this field specifies the start of the AAD data in the source buffer.

Definition at line 179 of file cpa_cy_sym_dp.h.

5.56.2.7 messageLenToHashInBytes

`Cpa32U _CpaCySymDpOpData::messageLenToHashInBytes`

The message length, in bytes, of the source buffer that the hash will be computed on.

Note

For CCM and GCM modes of operation, this value in this field is ignored, and the field is reserved for internal usage. The fields additionalAuthData and pAdditionalAuthData should be set instead.

For AES-GMAC (CPA_CY_SYM_HASH_AES_GMAC) mode of operation, this field specifies the length of the AAD data in the source buffer.

On some implementations, this length may be limited to a 16-bit value (65535 bytes).

Definition at line 192 of file cpa_cy_sym_dp.h.

5.56.2.8 additionalAuthData

`CpaPhysicalAddr _CpaCySymDpOpData::additionalAuthData`

Physical address of the Additional Authenticated Data (AAD), which is needed for authenticated cipher mechanisms (CCM and GCM), and to the IV for SNOW3G authentication (CPA_CY_SYM_HASH_SNOW3G_UIA2). For other authentication mechanisms, this value is ignored, and the field is reserved for internal usage.

The length of the data pointed to by this field is set up for the session in the CpaCySymHashAuthModeSetupData structure as part of the cpaCySymDpInitSession function call. This length must not exceed 240 bytes.

If AAD is not used, this address must be set to zero.

Specifically for CCM (CPA_CY_SYM_HASH_AES_CCM) and GCM (CPA_CY_SYM_HASH_AES_GCM), the caller should be setup as described in the same way as the corresponding field, pAdditionalAuthData, on the "traditional" API (see the CpaCySymOpData).

Note

For AES-GMAC (CPA_CY_SYM_HASH_AES_GMAC) mode of operation, this field is not used and should be set to 0. Instead the AAD data should be placed in the source buffer.

Definition at line 208 of file cpa_cy_sym_dp.h.

5.56.2.9 digestResult

`CpaPhysicalAddr _CpaCySymDpOpData::digestResult`

If the digestIsAppended member of the CpaCySymSessionSetupData structure is NOT set then this is the physical address of the location where the digest result should be inserted (in the case of digest generation) or where the purported digest exists (in the case of digest verification).

At session registration time, the client specified the digest result length with the digestResultLenInBytes member of the CpaCySymHashSetupData structure. The client must allocate at least digestResultLenInBytes of physically contiguous memory at this location.

For digest generation, the digest result will overwrite any data at this location.

Note

For GCM (CPA_CY_SYM_HASH_AES_GCM), for "digest result" read "authentication tag T".

If the digestIsAppended member of the CpaCySymSessionSetupData structure is set then this value is ignored and the digest result is understood to be in the destination buffer for digest generation, and in the source buffer for digest verification. The location of the digest result in this case is immediately following the region over which the digest is computed.

Definition at line 233 of file cpa_cy_sym_dp.h.

5.56.2.10 instanceHandle

`CpaInstanceHandle _CpaCySymDpOpData::instanceHandle`

Instance to which the request is to be enqueued.

Note

A callback function must have been registered on the instance using cpaCySymDpRegCbFunc.

Definition at line 259 of file cpa_cy_sym_dp.h.

5.56.2.11 sessionCtx

`CpaCySymDpSessionCtx _CpaCySymDpOpData::sessionCtx`

Session context specifying the cryptographic parameters for this request.

Note

The session must have been created using cpaCySymDpInitSession.

Definition at line 264 of file cpa_cy_sym_dp.h.

5.56.2.12 ivLenInBytes

`Cpa32U _CpaCySymDpOpData::ivLenInBytes`

Length of valid IV data pointed to by the pIv parameter.

- For block ciphers in CBC mode, or for Kasumi in F8 mode, or for SNOW3G in UEA2 mode, this is the length of the IV (which must be the same as the block length of the cipher).
- For block ciphers in CTR mode, this is the length of the counter (which must be the same as the block length of the cipher).
- For GCM mode, this is either 12 (for 96-bit IVs) or 16, in which case pIv points to J0.
- For CCM mode, this is the length of the nonce, which can be in the range 7 to 13 inclusive.

Definition at line 270 of file cpa_cy_sym_dp.h.

5.56.2.13 srcBuffer

`CpaPhysicalAddr _CpaCySymDpOpData::srcBuffer`

Physical address of the source buffer on which to operate. This is either:

- The location of the data, of length srcBufferLen; or,
- If srcBufferLen has the special value CPA_DP_BUFLIST, then srcBuffer contains the location where a CpaPhysBufferList is stored. In this case, the CpaPhysBufferList MUST be aligned on an 8-byte boundary.
- For optimum performance, the buffer should only contain the data region that the cryptographic operation(s) must be performed on. Any additional data in the source buffer may be copied to the destination buffer and this copy may degrade performance.

Definition at line 283 of file cpa_cy_sym_dp.h.

5.56.2.14 srcBufferLen

`Cpa32U _CpaCySymDpOpData::srcBufferLen`

Length of source buffer, or CPA_DP_BUFLIST.

Definition at line 297 of file cpa_cy_sym_dp.h.

5.56.2.15 dstBuffer

`CpaPhysicalAddr _CpaCySymDpOpData::dstBuffer`

Physical address of the destination buffer on which to operate. This is either:

- The location of the data, of length srcBufferLen; or,
- If srcBufferLen has the special value CPA_DP_BUFLIST, then srcBuffer contains the location where a CpaPhysBufferList is stored. In this case, the CpaPhysBufferList MUST be aligned on an 8-byte boundary.

For "in-place" operation, the dstBuffer may be identical to the srcBuffer.

Definition at line 299 of file cpa_cy_sym_dp.h.

5.56.2.16 dstBufferLen

`Cpa32U _CpaCySymDpOpData::dstBufferLen`

Length of destination buffer, or CPA_DP_BUFLIST.

Definition at line 312 of file cpa_cy_sym_dp.h.

5.56.2.17 thisPhys

`CpaPhysicalAddr _CpaCySymDpOpData::thisPhys`

Physical address of this data structure

Definition at line 315 of file cpa_cy_sym_dp.h.

5.56.2.18 pIv

`Cpa8U* _CpaCySymDpOpData::pIv`

Pointer to (and therefore, the virtual address of) the IV field above. Needed here because the driver in some cases writes to this field, in addition to sending it to the accelerator.

Definition at line 318 of file cpa_cy_sym_dp.h.

5.56.2.19 pAdditionalAuthData

`Cpa8U* _CpaCySymDpOpData::pAdditionalAuthData`

Pointer to (and therefore, the virtual address of) the additionalAuthData field above. Needed here because the driver in some cases writes to this field, in addition to sending it to the accelerator.

Definition at line 324 of file cpa_cy_sym_dp.h.

5.56.2.20 pCallbackTag

`void* _CpaCySymDpOpData::pCallbackTag`

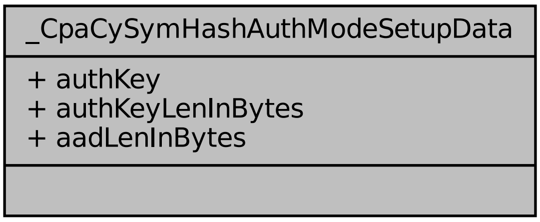
Opaque data that will be returned to the client in the function completion callback.

This opaque data is not used by the implementation of the API, but is simply returned as part of the asynchronous response. It may be used to store information that might be useful when processing the response later.

Definition at line 330 of file cpa_cy_sym_dp.h.

5.57 _CpaCySymHashAuthModeSetupData Struct Reference

Collaboration diagram for _CpaCySymHashAuthModeSetupData:



Data Fields

- Cpa8U ∗ authKey
- Cpa32U authKeyLenInBytes
- Cpa32U aadLenInBytes

5.57.1 Detailed Description

Hash Auth Mode Setup Data.

Description:

This structure contains data relating to a hash session in CPA_CY_SYM_HASH_MODE_AUTH mode.

Definition at line 444 of file cpa_cy_sym.h.

5.57.2 Field Documentation

5.57.2.1 authKey

`Cpa8U* _CpaCySymHashAuthModeSetupData::authKey`

Authentication key pointer. For the GCM (CPA_CY_SYM_HASH_AES_GCM) and CCM (CPA_CY_SYM_HASH_AES_CCM) modes of operation, this field is ignored; the authentication key is the same as the cipher key (see the field pCipherKey in struct CpaCySymCipherSetupData).

Definition at line 445 of file cpa_cy_sym.h.

5.57.2.2 authKeyLenInBytes

`Cpa32U _CpaCySymHashAuthModeSetupData::authKeyLenInBytes`

Length of the authentication key in bytes. The key length MUST be less than or equal to the block size of the algorithm. It is the client's responsibility to ensure that the key length is compliant with the standard being used (for example RFC 2104, FIPS 198a).

For the GCM (CPA_CY_SYM_HASH_AES_GCM) and CCM (CPA_CY_SYM_HASH_AES_CCM) modes of operation, this field is ignored; the authentication key is the same as the cipher key, and so is its length (see the field cipherKeyLenInBytes in struct CpaCySymCipherSetupData).

Definition at line 452 of file cpa_cy_sym.h.

5.57.2.3 aadLenInBytes

`Cpa32U _CpaCySymHashAuthModeSetupData::aadLenInBytes`

The length of the additional authenticated data (AAD) in bytes. The maximum permitted value is 240 bytes, unless otherwise specified below.

This field must be specified when the hash algorithm is one of the following:

- For SNOW3G (CPA_CY_SYM_HASH_SNOW3G_UIA2), this is the length of the IV (which should be 16).
- For GCM (CPA_CY_SYM_HASH_AES_GCM). In this case, this is the length of the Additional Authenticated Data (called A, in NIST SP800-38D).
- For CCM (CPA_CY_SYM_HASH_AES_CCM). In this case, this is the length of the associated data (called A, in NIST SP800-38C). Note that this does NOT include the length of any padding, or the 18 bytes reserved at the start of the above field to store the block B0 and the encoded length. The maximum permitted value in this case is 222 bytes.

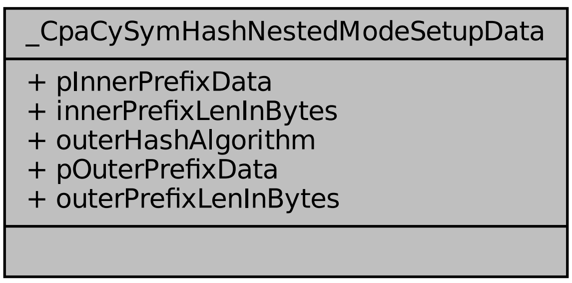
  Note

  For AES-GMAC (CPA_CY_SYM_HASH_AES_GMAC) mode of operation this field is not used and should be set to 0. Instead the length of the AAD data is specified in the messageLenToHashInBytes field of the CpaCySymOpData structure.

Definition at line 464 of file cpa_cy_sym.h.

5.58 _CpaCySymHashNestedModeSetupData Struct Reference

Collaboration diagram for _CpaCySymHashNestedModeSetupData:



Data Fields

- Cpa8U * pInnerPrefixData
- Cpa32U innerPrefixLenInBytes
- CpaCySymHashAlgorithm outerHashAlgorithm
- Cpa8U * pOuterPrefixData
- Cpa32U outerPrefixLenInBytes

5.58.1 Detailed Description

Hash Mode Nested Setup Data.

Description:

This structure contains data relating to a hash session in CPA_CY_SYM_HASH_MODE_NESTED mode.

Definition at line 413 of file cpa_cy_sym.h.

5.58.2 Field Documentation

5.58.2.1 pInnerPrefixData

`Cpa8U* _CpaCySymHashNestedModeSetupData::pInnerPrefixData`

A pointer to a buffer holding the Inner Prefix data. For optimal performance the prefix data SHOULD be 8-byte aligned. This data is prepended to the data being hashed before the inner hash operation is performed.

Definition at line 414 of file cpa_cy_sym.h.

5.58.2.2 innerPrefixLenInBytes

`Cpa32U _CpaCySymHashNestedModeSetupData::innerPrefixLenInBytes`

The inner prefix length in bytes. The maximum size the prefix data can be is 255 bytes.

Definition at line 419 of file cpa_cy_sym.h.

5.58.2.3 outerHashAlgorithm

`CpaCySymHashAlgorithm _CpaCySymHashNestedModeSetupData::outerHashAlgorithm`

The hash algorithm used for the outer hash. Note: The inner hash algorithm is provided in the hash context.

Definition at line 422 of file cpa_cy_sym.h.

5.58.2.4 pOuterPrefixData

`Cpa8U* _CpaCySymHashNestedModeSetupData::pOuterPrefixData`

A pointer to a buffer holding the Outer Prefix data. For optimal performance the prefix data SHOULD be 8-byte aligned. This data is prepended to the output from the inner hash operation before the outer hash operation is performed.

Definition at line 425 of file cpa_cy_sym.h.

5.58.2.5 outerPrefixLenInBytes

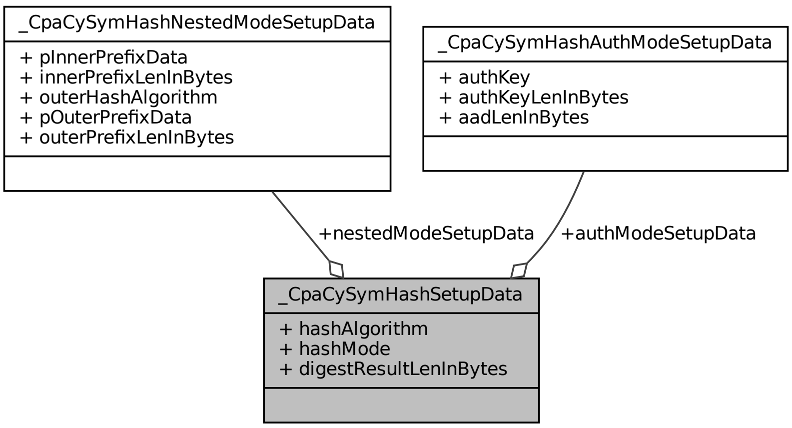
`Cpa32U _CpaCySymHashNestedModeSetupData::outerPrefixLenInBytes`

The outer prefix length in bytes. The maximum size the prefix data can be is 255 bytes.

Definition at line 430 of file cpa_cy_sym.h.

5.59 _CpaCySymHashSetupData Struct Reference

Collaboration diagram for _CpaCySymHashSetupData:



Data Fields

- CpaCySymHashAlgorithm hashAlgorithm
- CpaCySymHashMode hashMode
- Cpa32U digestResultLenInBytes
- CpaCySymHashAuthModeSetupData authModeSetupData
- CpaCySymHashNestedModeSetupData nestedModeSetupData

5.59.1 Detailed Description

Hash Setup Data.

Description:

This structure contains data relating to a hash session. The fields hashAlgorithm, hashMode and digestResultLenInBytes are common to all three hash modes and MUST be set for each mode.

Definition at line 501 of file cpa_cy_sym.h.

5.59.2 Field Documentation

5.59.2.1 hashAlgorithm

`CpaCySymHashAlgorithm _CpaCySymHashSetupData::hashAlgorithm`

Hash algorithm. For mode CPA_CY_SYM_MODE_HASH_NESTED, this is the inner hash algorithm.

Definition at line 502 of file cpa_cy_sym.h.

5.59.2.2 hashMode

`CpaCySymHashMode _CpaCySymHashSetupData::hashMode`

Mode of the hash operation. Valid options include plain, auth or nested hash mode.

Definition at line 505 of file cpa_cy_sym.h.

5.59.2.3 digestResultLenInBytes

`Cpa32U _CpaCySymHashSetupData::digestResultLenInBytes`

Length of the digest to be returned. If the verify option is set, this specifies the length of the digest to be compared for the session.

For CCM (CPA_CY_SYM_HASH_AES_CCM), this is the octet length of the MAC, which can be one of 4, 6, 8, 10, 12, 14 or 16.

For GCM (CPA_CY_SYM_HASH_AES_GCM), this is the length in bytes of the authentication tag.

If the value is less than the maximum length allowed by the hash, the result shall be truncated. If the value is greater than the maximum length allowed by the hash, an error (CPA_STATUS_INVALID_PARAM) is returned from the function cpaCySymInitSession.

In the case of nested hash, it is the outer hash which determines the maximum length allowed.

Definition at line 508 of file cpa_cy_sym.h.

5.59.2.4 authModeSetupData

`CpaCySymHashAuthModeSetupData _CpaCySymHashSetupData::authModeSetupData`

Authentication Mode Setup Data. Only valid for mode CPA_CY_SYM_MODE_HASH_AUTH

Definition at line 527 of file cpa_cy_sym.h.

5.59.2.5 nestedModeSetupData

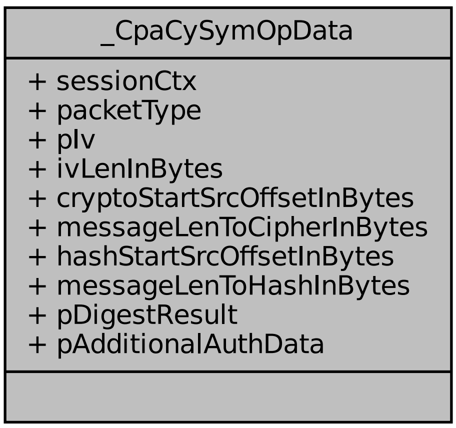
`CpaCySymHashNestedModeSetupData _CpaCySymHashSetupData::nestedModeSetupData`

Nested Hash Mode Setup Data Only valid for mode CPA_CY_SYM_MODE_HASH_NESTED

Definition at line 530 of file cpa_cy_sym.h.

5.60 _CpaCySymOpData Struct Reference

Collaboration diagram for _CpaCySymOpData:



Data Fields

- CpaCySymSessionCtx sessionCtx
- CpaCySymPacketType packetType
- Cpa8U * pIv
- Cpa32U ivLenInBytes
- Cpa32U cryptoStartSrcOffsetInBytes
- Cpa32U messageLenToCipherInBytes
- Cpa32U hashStartSrcOffsetInBytes
- Cpa32U messageLenToHashInBytes
- Cpa8U * pDigestResult
- Cpa8U * pAdditionalAuthData

5.60.1 Detailed Description

Cryptographic Component Operation Data.

Description:

This structure contains data relating to performing cryptographic processing on a data buffer. This request is used with cpaCySymPerformOp() call for performing cipher, hash, auth cipher or a combined hash and cipher operation.

See also

CpaCySymPacketType

Note

If the client modifies or frees the memory referenced in this structure after it has been submitted to the cpaCySymPerformOp function, and before it has been returned in the callback, undefined behavior will result.

Definition at line 737 of file cpa_cy_sym.h.

5.60.2 Field Documentation

5.60.2.1 sessionCtx

`CpaCySymSessionCtx _CpaCySymOpData::sessionCtx`

Handle for the initialized session context

Definition at line 738 of file cpa_cy_sym.h.

5.60.2.2 packetType

`CpaCySymPacketType _CpaCySymOpData::packetType`

Selects the packet type

Definition at line 740 of file cpa_cy_sym.h.

5.60.2.3 pIv

`Cpa8U* _CpaCySymOpData::pIv`

Initialization Vector or Counter.

- For block ciphers in CBC or F8 mode, or for Kasumi in F8 mode, or for SNOW3G in UEA2 mode, this is the Initialization Vector (IV) value.
- For block ciphers in CTR mode, this is the counter.
- For GCM mode, this is either the IV (if the length is 96 bits) or J0 (for other sizes), where J0 is as defined by NIST SP800-38D. Regardless of the IV length, a full 16 bytes needs to be allocated.
- For CCM mode, the first byte is reserved, and the nonce should be written starting at &pIv[1] (to allow space for the implementation to write in the flags in the first byte). Note that a full 16 bytes should be allocated, even though the ivLenInBytes field will have a value less than this. The macro CPA_CY_SYM_CCM_SET_NONCE may be used here.
- For AES-XTS, this is the 128bit tweak, i, from IEEE Std 1619-2007.

For optimum performance, the data pointed to SHOULD be 8-byte aligned.

The IV/Counter will be updated after every partial cryptographic operation.

Definition at line 742 of file cpa_cy_sym.h.

5.60.2.4 ivLenInBytes

`Cpa32U _CpaCySymOpData::ivLenInBytes`

Length of valid IV data pointed to by the pIv parameter.

- For block ciphers in CBC or F8 mode, or for Kasumi in F8 mode, or for SNOW3G in UEA2 mode, this is the length of the IV (which must be the same as the block length of the cipher).
- For block ciphers in CTR mode, this is the length of the counter (which must be the same as the block length of the cipher).
- For GCM mode, this is either 12 (for 96-bit IVs) or 16, in which case pIv points to J0.
- For CCM mode, this is the length of the nonce, which can be in the range 7 to 13 inclusive.

Definition at line 766 of file cpa_cy_sym.h.

5.60.2.5 cryptoStartSrcOffsetInBytes

`Cpa32U _CpaCySymOpData::cryptoStartSrcOffsetInBytes`

Starting point for cipher processing, specified as number of bytes from start of data in the source buffer. The result of the cipher operation will be written back into the output buffer starting at this location.

Definition at line 779 of file cpa_cy_sym.h.

5.60.2.6 messageLenToCipherInBytes

`Cpa32U _CpaCySymOpData::messageLenToCipherInBytes`

The message length, in bytes, of the source buffer on which the cryptographic operation will be computed. This must be a multiple of the block size if a block cipher is being used. This is also the same as the result length.

Note

In the case of CCM (CPA_CY_SYM_HASH_AES_CCM), this value should not include the length of the padding or the length of the MAC; the driver will compute the actual number of bytes over which the encryption will occur, which will include these values.

There are limitations on this length for partial operations. Refer to the cpaCySymPerformOp function description for details.

On some implementations, this length may be limited to a 16-bit value (65535 bytes).

For AES-GMAC (CPA_CY_SYM_HASH_AES_GMAC), this field should be set to 0.

Definition at line 785 of file cpa_cy_sym.h.

5.60.2.7 hashStartSrcOffsetInBytes

`Cpa32U _CpaCySymOpData::hashStartSrcOffsetInBytes`

Starting point for hash processing, specified as number of bytes from start of packet in source buffer.

Note

For CCM and GCM modes of operation, this field is ignored. The field pAdditionalAuthData field should be set instead.

For AES-GMAC (CPA_CY_SYM_HASH_AES_GMAC) mode of operation, this field specifies the start of the AAD data in the source buffer.

Definition at line 806 of file cpa_cy_sym.h.

5.60.2.8 messageLenToHashInBytes

`Cpa32U _CpaCySymOpData::messageLenToHashInBytes`

The message length, in bytes, of the source buffer that the hash will be computed on.

Note

There are limitations on this length for partial operations. Refer to the cpaCySymPerformOp function description for details.

For CCM and GCM modes of operation, this field is ignored. The field pAdditionalAuthData field should be set instead.

For AES-GMAC (CPA_CY_SYM_HASH_AES_GMAC) mode of operation, this field specifies the length of the AAD data in the source buffer. The maximum length supported for AAD data for AES-GMAC is 16383 bytes.

On some implementations, this length may be limited to a 16-bit value (65535 bytes).

Definition at line 817 of file cpa_cy_sym.h.

5.60.2.9 pDigestResult

`Cpa8U* _CpaCySymOpData::pDigestResult`

If the digestIsAppended member of the CpaCySymSessionSetupData structure is NOT set then this is a pointer to the location where the digest result should be inserted (in the case of digest generation) or where the purported digest exists (in the case of digest verification).

At session registration time, the client specified the digest result length with the digestResultLenInBytes member of the CpaCySymHashSetupData structure. The client must allocate at least digestResultLenInBytes of physically contiguous memory at this location.

For partial packet processing without algorithm chaining, this pointer will be ignored for all but the final partial operation.

For digest generation, the digest result will overwrite any data at this location.

Note

For GCM (CPA_CY_SYM_HASH_AES_GCM), for "digest result" read "authentication tag T".

If the digestIsAppended member of the CpaCySymSessionSetupData structure is set then this value is ignored and the digest result is understood to be in the destination buffer for digest generation, and in the source buffer for digest verification. The location of the digest result in this case is immediately following the region over which the digest is computed.

Definition at line 835 of file cpa_cy_sym.h.

5.60.2.10 pAdditionalAuthData

`Cpa8U* _CpaCySymOpData::pAdditionalAuthData`

Pointer to Additional Authenticated Data (AAD) needed for authenticated cipher mechanisms (CCM and GCM), and to the IV for SNOW3G authentication (CPA_CY_SYM_HASH_SNOW3G_UIA2). For other authentication mechanisms this pointer is ignored.

The length of the data pointed to by this field is set up for the session in the CpaCySymHashAuthModeSetupData structure as part of the cpaCySymInitSession function call. This length must not exceed 240 bytes.

Specifically for CCM (CPA_CY_SYM_HASH_AES_CCM), the caller should setup this field as follows:

- the nonce should be written starting at an offset of one byte into the array, leaving room for the implementation to write in the flags to the first byte. For example,
  memcpy(&pOpData->pAdditionalAuthData[1], pNonce, nonceLen);
  The macro CPA_CY_SYM_CCM_SET_NONCE may be used here.
- the additional authentication data itself should be written starting at an offset of 18 bytes into the array, leaving room for the length encoding in the first two bytes of the second block. For example,
  memcpy(&pOpData->pAdditionalAuthData[18], pAad, aadLen);
  The macro CPA_CY_SYM_CCM_SET_AAD may be used here.
- the array should be big enough to hold the above fields, plus any padding to round this up to the nearest multiple of the block size (16 bytes). Padding will be added by the implementation.

Finally, for GCM (CPA_CY_SYM_HASH_AES_GCM), the caller should setup this field as follows:

- the AAD is written in starting at byte 0
- the array must be big enough to hold the AAD, plus any padding to round this up to the nearest multiple of the block size (16 bytes). Padding will be added by the implementation.

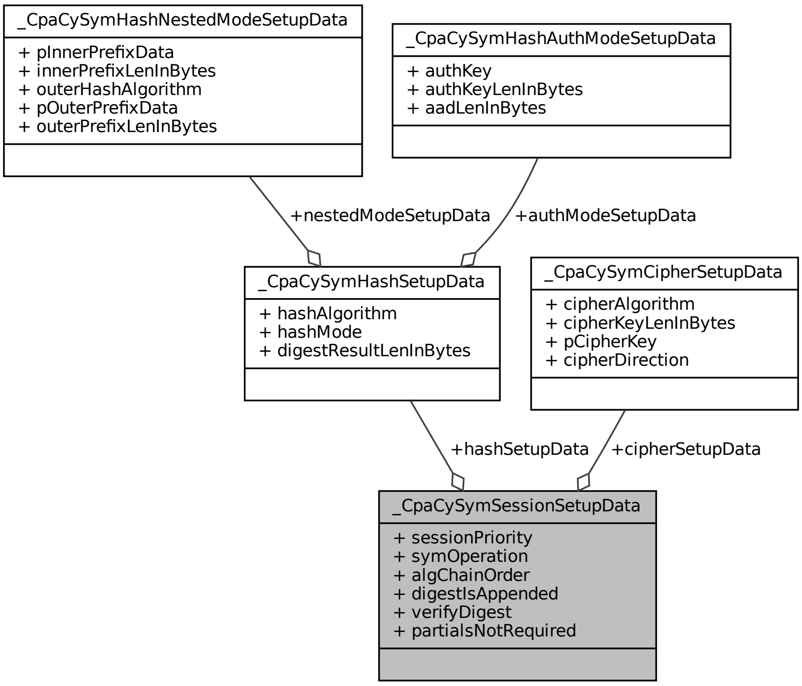
Note

For AES-GMAC (CPA_CY_SYM_HASH_AES_GMAC) mode of operation, this field is not used and should be set to 0. Instead the AAD data should be placed in the source buffer.

Definition at line 863 of file cpa_cy_sym.h.

5.61 _CpaCySymSessionSetupData Struct Reference

Collaboration diagram for _CpaCySymSessionSetupData:



Data Fields

- CpaCyPriority sessionPriority
- CpaCySymOp symOperation
- CpaCySymCipherSetupData cipherSetupData
- CpaCySymHashSetupData hashSetupData
- CpaCySymAlgChainOrder algChainOrder
- CpaBoolean digestIsAppended
- CpaBoolean verifyDigest
- CpaBoolean partialsNotRequired

5.61.1 Detailed Description

Session Setup Data.

Description:

This structure contains data relating to setting up a session. The client needs to complete the information in this structure in order to setup a session.

Definition at line 624 of file cpa_cy_sym.h.

5.61.2 Field Documentation

5.61.2.1 sessionPriority

`CpaCyPriority _CpaCySymSessionSetupData::sessionPriority`

Priority of this session

Definition at line 625 of file cpa_cy_sym.h.

5.61.2.2 symOperation

`CpaCySymOp _CpaCySymSessionSetupData::symOperation`

Operation to perfom

Definition at line 627 of file cpa_cy_sym.h.

5.61.2.3 cipherSetupData

`CpaCySymCipherSetupData _CpaCySymSessionSetupData::cipherSetupData`

Cipher Setup Data for the session. This member is ignored for the CPA_CY_SYM_OP_HASH operation.

Definition at line 629 of file cpa_cy_sym.h.

5.61.2.4 hashSetupData

`CpaCySymHashSetupData _CpaCySymSessionSetupData::hashSetupData`

Hash Setup Data for a session. This member is ignored for the CPA_CY_SYM_OP_CIPHER operation.

Definition at line 632 of file cpa_cy_sym.h.

5.61.2.5 algChainOrder

`CpaCySymAlgChainOrder _CpaCySymSessionSetupData::algChainOrder`

If this operation data structure relates to an algorithm chaining session then this parameter determines the order in which the chained operations are performed. If this structure does not relate to an algorithm chaining session then this parameter will be ignored.

Note

In the case of authenticated ciphers (GCM and CCM), which are also presented as "algorithm chaining", this value is also ignored. The chaining order is defined by the authenticated cipher, in those cases.

Definition at line 635 of file cpa_cy_sym.h.

5.61.2.6 digestIsAppended

`CpaBoolean _CpaCySymSessionSetupData::digestIsAppended`

Flag indicating whether the digest is appended immediately following the region over which the digest is computed. This is true for both IPsec packets and SSL/TLS records.

If this flag is set, then the value of the pDigestResult field of the structure CpaCySymOpData is ignored.

Note

The value of this field is ignored for the authenticated cipher AES_CCM as the digest must be appended in this case.

Setting digestIsAppended for hash only operations when verifyDigest is also set is not supported. For hash only operations when verifyDigest is set, digestIsAppended should be set to CPA_FALSE.

Definition at line 645 of file cpa_cy_sym.h.

5.61.2.7 verifyDigest

`CpaBoolean _CpaCySymSessionSetupData::verifyDigest`

This flag is relevant only for operations which generate a message digest. If set to true, the computed digest will not be written back to the buffer location specified by other parameters, but instead will be verified (i.e. compared to the value passed in at that location). The number of bytes to be written or compared is indicated by the digest output length for the session.

Note

> This option is only valid for full packets and for final partial packets when using partials without algorithm chaining.
>
> The value of this field is ignored for the authenticated ciphers (AES_CCM and AES_GCM). Digest verification is always done for these (when the direction is decrypt) and unless the DP API is used, the message buffer will be zeroed if verification fails. When using the DP API, it is the API clients responsibility to clear the message buffer when digest verification fails.

Definition at line 660 of file cpa_cy_sym.h.

5.61.2.8 partialsNotRequired

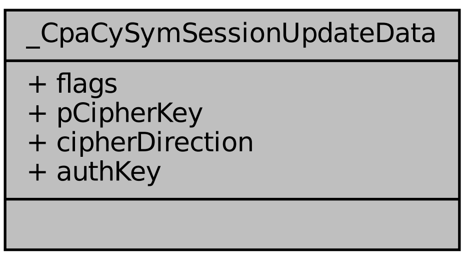
`CpaBoolean _CpaCySymSessionSetupData::partialsNotRequired`

This flag indicates if partial packet processing is required for this session. If set to true, partial packet processing will not be enabled for this session and any calls to cpaCySymPerformOp() with the packetType parameter set to a value other than CPA_CY_SYM_PACKET_TYPE_FULL will fail.

Definition at line 676 of file cpa_cy_sym.h.

5.62 _CpaCySymSessionUpdateData Struct Reference

Collaboration diagram for _CpaCySymSessionUpdateData:



Data Fields

- Cpa32U flags
- Cpa8U * pCipherKey
- CpaCySymCipherDirection cipherDirection
- Cpa8U * authKey

5.62.1 Detailed Description

Session Update Data.

Description:

This structure contains data relating to resetting a session.

Definition at line 692 of file cpa_cy_sym.h.

5.62.2 Field Documentation

5.62.2.1 flags

`Cpa32U _CpaCySymSessionUpdateData::flags`

Flags indicating which fields to update. All bits should be set to 0 except those fields to be updated.

Definition at line 693 of file cpa_cy_sym.h.

5.62.2.2 pCipherKey

`Cpa8U* _CpaCySymSessionUpdateData::pCipherKey`

Cipher key. The same restrictions apply as described in the corresponding field of the data structure CpaCySymCipherSetupData.

Definition at line 700 of file cpa_cy_sym.h.

5.62.2.3 cipherDirection

`CpaCySymCipherDirection _CpaCySymSessionUpdateData::cipherDirection`

This parameter determines if the cipher operation is an encrypt or a decrypt operation. The same restrictions apply as described in the corresponding field of the data structure CpaCySymCipherSetupData.

Definition at line 705 of file cpa_cy_sym.h.

5.62.2.4 authKey

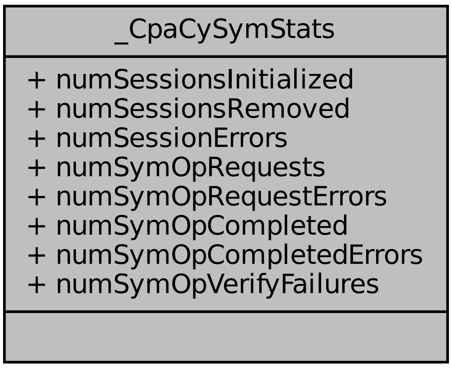
`Cpa8U* _CpaCySymSessionUpdateData::authKey`

Authentication key pointer. The same restrictions apply as described in the corresponding field of the data structure CpaCySymHashAuthModeSetupData.

Definition at line 711 of file cpa_cy_sym.h.

5.63 _CpaCySymStats Struct Reference

Collaboration diagram for _CpaCySymStats:



Data Fields

- Cpa32U numSessionsInitialized
- Cpa32U numSessionsRemoved
- Cpa32U numSessionErrors
- Cpa32U numSymOpRequests
- Cpa32U numSymOpRequestErrors
- Cpa32U numSymOpCompleted
- Cpa32U numSymOpCompletedErrors
- Cpa32U numSymOpVerifyFailures

5.63.1 Detailed Description

Cryptographic Component Statistics.

Deprecated As of v1.3 of the cryptographic API, this structure has been deprecated, replaced by CpaCySymStats64.

Description:

This structure contains statistics on the Symmetric Cryptographic operations. Statistics are set to zero when the component is initialized.

Definition at line 953 of file cpa_cy_sym.h.

5.63.2 Field Documentation

5.63.2.1 numSessionsInitialized

`Cpa32U _CpaCySymStats::numSessionsInitialized`

Number of session initialized

Definition at line 954 of file cpa_cy_sym.h.

5.63.2.2 numSessionsRemoved

`Cpa32U _CpaCySymStats::numSessionsRemoved`

Number of sessions removed

Definition at line 956 of file cpa_cy_sym.h.

5.63.2.3 numSessionErrors

`Cpa32U _CpaCySymStats::numSessionErrors`

Number of session initialized and removed errors.

Definition at line 958 of file cpa_cy_sym.h.

5.63.2.4 numSymOpRequests

`Cpa32U _CpaCySymStats::numSymOpRequests`

Number of successful symmetric operation requests.

Definition at line 960 of file cpa_cy_sym.h.

5.63.2.5 numSymOpRequestErrors

`Cpa32U _CpaCySymStats::numSymOpRequestErrors`

Number of operation requests that had an error and could not be processed.

Definition at line 962 of file cpa_cy_sym.h.

5.63.2.6 numSymOpCompleted

`Cpa32U _CpaCySymStats::numSymOpCompleted`

Number of operations that completed successfully.

Definition at line 965 of file cpa_cy_sym.h.

5.63.2.7 numSymOpCompletedErrors

`Cpa32U _CpaCySymStats::numSymOpCompletedErrors`

Number of operations that could not be completed successfully due to errors.

Definition at line 967 of file cpa_cy_sym.h.

5.63.2.8 numSymOpVerifyFailures

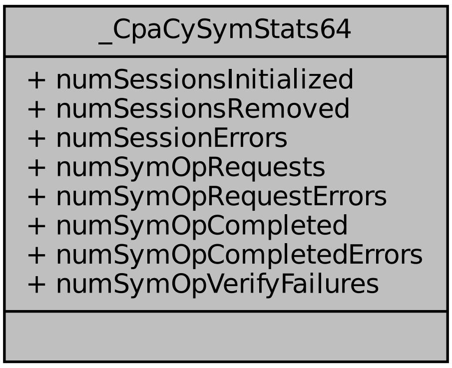
`Cpa32U _CpaCySymStats::numSymOpVerifyFailures`

Number of operations that completed successfully, but the result of the digest verification test was that it failed. Note that this does not indicate an error condition.

Definition at line 970 of file cpa_cy_sym.h.

5.64 _CpaCySymStats64 Struct Reference

Collaboration diagram for _CpaCySymStats64:



Data Fields

- Cpa64U numSessionsInitialized
- Cpa64U numSessionsRemoved
- Cpa64U numSessionErrors
- Cpa64U numSymOpRequests
- Cpa64U numSymOpRequestErrors
- Cpa64U numSymOpCompleted
- Cpa64U numSymOpCompletedErrors
- Cpa64U numSymOpVerifyFailures

5.64.1 Detailed Description

Cryptographic Component Statistics (64-bit version).

Description:

This structure contains a 64-bit version of the statistics on the Symmetric Cryptographic operations. Statistics are set to zero when the component is initialized.

Definition at line 985 of file cpa_cy_sym.h.

5.64.2 Field Documentation

5.64.2.1 numSessionsInitialized

`Cpa64U _CpaCySymStats64::numSessionsInitialized`

Number of session initialized

Definition at line 986 of file cpa_cy_sym.h.

5.64.2.2 numSessionsRemoved

`Cpa64U _CpaCySymStats64::numSessionsRemoved`

Number of sessions removed

Definition at line 988 of file cpa_cy_sym.h.

5.64.2.3 numSessionErrors

```
Cpa64U _CpaCySymStats64::numSessionErrors
```

Number of session initialized and removed errors.

Definition at line 990 of file cpa_cy_sym.h.

5.64.2.4 numSymOpRequests

```
Cpa64U _CpaCySymStats64::numSymOpRequests
```

Number of successful symmetric operation requests.

Definition at line 992 of file cpa_cy_sym.h.

5.64.2.5 numSymOpRequestErrors

```
Cpa64U _CpaCySymStats64::numSymOpRequestErrors
```

Number of operation requests that had an error and could not be processed.

Definition at line 994 of file cpa_cy_sym.h.

5.64.2.6 numSymOpCompleted

```
Cpa64U _CpaCySymStats64::numSymOpCompleted
```

Number of operations that completed successfully.

Definition at line 997 of file cpa_cy_sym.h.

5.64.2.7 numSymOpCompletedErrors

```
Cpa64U _CpaCySymStats64::numSymOpCompletedErrors
```

Number of operations that could not be completed successfully due to errors.

Definition at line 999 of file cpa_cy_sym.h.

5.64.2.8 numSymOpVerifyFailures

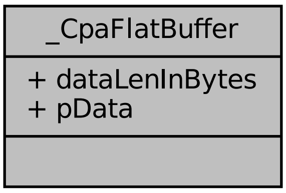
`Cpa64U _CpaCySymStats64::numSymOpVerifyFailures`

Number of operations that completed successfully, but the result of the digest verification test was that it failed. Note that this does not indicate an error condition.

Definition at line 1002 of file cpa_cy_sym.h.

5.65 _CpaFlatBuffer Struct Reference

Collaboration diagram for _CpaFlatBuffer:



Data Fields

- Cpa32U dataLenInBytes
- Cpa8U ∗ pData

5.65.1 Detailed Description

Flat buffer structure containing a pointer and length member.

Description:

A flat buffer structure. The data pointer, pData, is a virtual address. An API instance may require the actual data to be in contiguous physical memory as determined by CpaInstanceInfo2.

Definition at line 129 of file cpa.h.

5.65.2 Field Documentation

5.65.2.1 dataLenInBytes

```
Cpa32U _CpaFlatBuffer::dataLenInBytes
```

Data length specified in bytes. When used as an input parameter to a function, the length specifies the current length of the buffer. When used as an output parameter to a function, the length passed in specifies the maximum length of the buffer on return (i.e. the allocated length). The implementation will not write past this length. On return, the length is always unchanged.

Definition at line 130 of file cpa.h.

5.65.2.2 pData

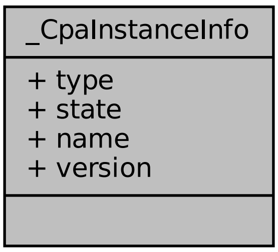
```
Cpa8U* _CpaFlatBuffer::pData
```

The data pointer is a virtual address, however the actual data pointed to is required to be in contiguous physical memory unless the field requiresPhysicallyContiguousMemory in CpaInstanceInfo2 is false.

Definition at line 138 of file cpa.h.

5.66 _CpaInstanceInfo Struct Reference

Collaboration diagram for _CpaInstanceInfo:



Data Fields

- enum _CpaInstanceType type
- enum _CpaInstanceState state
- Cpa8U name [CPA_INSTANCE_MAX_NAME_SIZE_IN_BYTES]
- Cpa8U version [CPA_INSTANCE_MAX_VERSION_SIZE_IN_BYTES]

5.66.1 Detailed Description

Instance Info Structure

Deprecated As of v1.3 of the Crypto API, this structure has been deprecated, replaced by CpaInstanceInfo2.

Description:

Structure that contains the information to describe the instance.

Definition at line 463 of file cpa.h.

5.66.2 Field Documentation

5.66.2.1 type

```
enum _CpaInstanceType _CpaInstanceInfo::type
```

Type definition for this instance.

Definition at line 464 of file cpa.h.

5.66.2.2 state

```
enum _CpaInstanceState _CpaInstanceInfo::state
```

Operational state of the instance.

Definition at line 466 of file cpa.h.

5.66.2.3 name

```
Cpa8U _CpaInstanceInfo::name[CPA_INSTANCE_MAX_NAME_SIZE_IN_BYTES]
```

Simple text string identifier for the instance.

Definition at line 468 of file cpa.h.

5.66.2.4 version

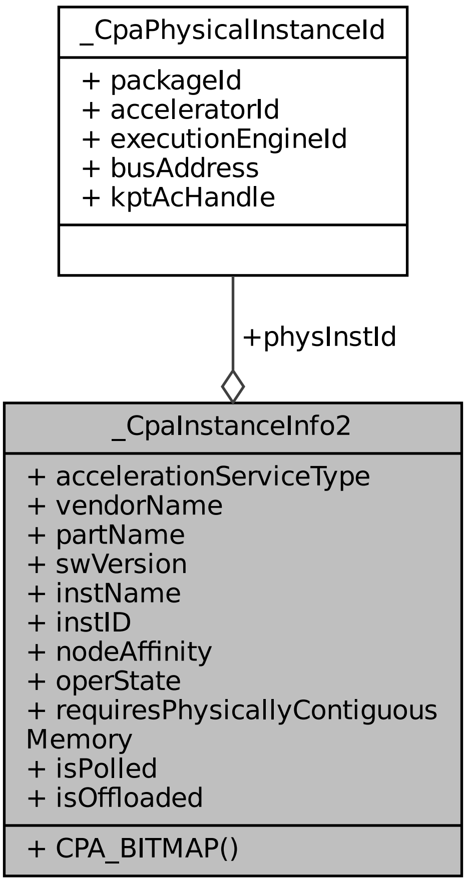
```
Cpa8U _CpaInstanceInfo::version[CPA_INSTANCE_MAX_VERSION_SIZE_IN_BYTES]
```

Version string. There may be multiple versions of the same type of instance accessible through a particular library.

Definition at line 470 of file cpa.h.

5.67 _CpaInstanceInfo2 Struct Reference

Collaboration diagram for _CpaInstanceInfo2:



Public Member Functions

- CPA_BITMAP (coreAffinity, CPA_MAX_CORES)

Data Fields

- CpaAccelerationServiceType accelerationServiceType
- Cpa8U vendorName [CPA_INST_VENDOR_NAME_SIZE]
- Cpa8U partName [CPA_INST_PART_NAME_SIZE]
- Cpa8U swVersion [CPA_INST_SW_VERSION_SIZE]
- Cpa8U instName [CPA_INST_NAME_SIZE]
- Cpa8U instID [CPA_INST_ID_SIZE]
- CpaPhysicalInstanceId physInstId
- Cpa32U nodeAffinity
- CpaOperationalState operState
- CpaBoolean requiresPhysicallyContiguousMemory
- CpaBoolean isPolled
- CpaBoolean isOffloaded

5.67.1 Detailed Description

Instance Info Structure, version 2

Description:

Structure that contains the information to describe the instance.

Definition at line 525 of file cpa.h.

5.67.2 Member Function Documentation

5.67.2.1 CPA_BITMAP()

```
_CpaInstanceInfo2::CPA_BITMAP (
            coreAffinity ,
            CPA_MAX_CORES  )
```

A bitmap identifying the core or cores to which the instance is affinitized in an SMP operating system.

The term core here is used to mean a "logical" core - for example, in a dual-processor, quad-core system with hyperthreading (two threads per core), there would be 16 such cores (2 processors x 4 cores/processor x 2 threads/core). The numbering of these cores and the corresponding bit positions is OS-specific. Note that Linux refers to this as "processor affinity" or "CPU affinity", and refers to the bitmap as a "cpumask".

The term "affinity" is used to mean that this is the core on which the callback function will be invoked when using the asynchronous mode of the API. In a hardware-based implementation of the API, this might be the core to which the interrupt is affinitized. In a software-based implementation, this might be the core to which the process running the algorithm is affinitized. Where there is no affinity, the bitmap can be set to all zeroes.

This bitmap should be manipulated using the macros CPA_BITMAP_BIT_SET, CPA_BITMAP_BIT_CLEAR and CPA_BITMAP_BIT_TEST.

5.67.3 Field Documentation

5.67.3.1 accelerationServiceType

`CpaAccelerationServiceType _CpaInstanceInfo2::accelerationServiceType`

Type of service provided by this instance.

Definition at line 526 of file cpa.h.

5.67.3.2 vendorName

`Cpa8U _CpaInstanceInfo2::vendorName[CPA_INST_VENDOR_NAME_SIZE]`

String identifying the vendor of the accelerator.

Definition at line 530 of file cpa.h.

5.67.3.3 partName

`Cpa8U _CpaInstanceInfo2::partName[CPA_INST_PART_NAME_SIZE]`

String identifying the part (name and/or number).

Definition at line 535 of file cpa.h.

5.67.3.4 swVersion

`Cpa8U _CpaInstanceInfo2::swVersion[CPA_INST_SW_VERSION_SIZE]`

String identifying the version of the software associated with the instance. For hardware-based implementations of the API, this should be the driver version. For software-based implementations of the API, this should be the version of the library.

Note that this should NOT be used to store the version of the API, nor should it be used to report the hardware revision (which can be captured as part of the partName, if required).

Definition at line 540 of file cpa.h.

5.67.3.5 instName

`Cpa8U _CpaInstanceInfo2::instName[CPA_INST_NAME_SIZE]`

String identifying the name of the instance.

Definition at line 553 of file cpa.h.

5.67.3.6 instID

`Cpa8U _CpaInstanceInfo2::instID[CPA_INST_ID_SIZE]`

String containing a unique identifier for the instance

Definition at line 557 of file cpa.h.

5.67.3.7 physInstId

`CpaPhysicalInstanceId _CpaInstanceInfo2::physInstId`

Identifies the "physical instance" of the accelerator.

Definition at line 560 of file cpa.h.

5.67.3.8 nodeAffinity

`Cpa32U _CpaInstanceInfo2::nodeAffinity`

Identifies the processor complex, or node, to which the accelerator is physically connected, to help identify locality in NUMA systems.

The values taken by this attribute will typically be in the range 0..n-1, where n is the number of nodes (processor complexes) in the system. For example, in a dual-processor configuration, n=2. The precise values and their interpretation are OS-specific.

Definition at line 589 of file cpa.h.

5.67.3.9 operState

`CpaOperationalState _CpaInstanceInfo2::operState`

Operational state of the instance.

Definition at line 598 of file cpa.h.

5.67.3.10 requiresPhysicallyContiguousMemory

`CpaBoolean _CpaInstanceInfo2::requiresPhysicallyContiguousMemory`

Specifies whether the data pointed to by flat buffers (CpaFlatBuffer::pData) supplied to this instance must be in physically contiguous memory.

Definition at line 600 of file cpa.h.

5.67.3.11 isPolled

`CpaBoolean _CpaInstanceInfo2::isPolled`

Specifies whether the instance must be polled, or is event driven. For hardware accelerators, the alternative to polling would be interrupts.

Definition at line 604 of file cpa.h.

5.67.3.12 isOffloaded

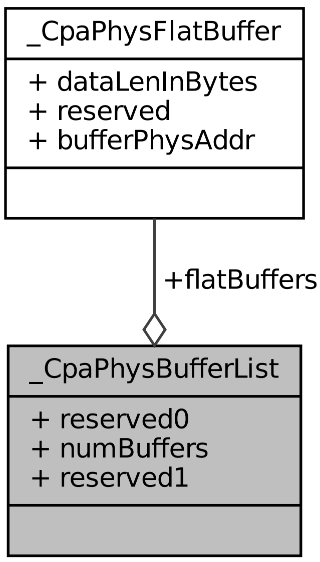
`CpaBoolean _CpaInstanceInfo2::isOffloaded`

Identifies whether the instance uses hardware offload, or is a software-only implementation.

Definition at line 608 of file cpa.h.

5.68 _CpaPhysBufferList Struct Reference

Collaboration diagram for _CpaPhysBufferList:



Data Fields

- Cpa64U reserved0
- Cpa32U numBuffers
- Cpa32U reserved1
- CpaPhysFlatBuffer flatBuffers [ ]

5.68.1 Detailed Description

Scatter/gather list containing an array of flat buffers with physical addresses.

Description:

Similar to CpaBufferList, this buffer structure is typically used to represent a region of memory which is not physically contiguous, by describing it as a collection of buffers, each of which is physically contiguous. The difference is that, in this case, the individual "flat" buffers are represented using physical, rather than virtual, addresses.

Definition at line 224 of file cpa.h.

5.68.2 Field Documentation

5.68.2.1 reserved0

`Cpa64U _CpaPhysBufferList::reserved0`

Reserved for internal usage

Definition at line 225 of file cpa.h.

5.68.2.2 numBuffers

`Cpa32U _CpaPhysBufferList::numBuffers`

Number of buffers in the list

Definition at line 227 of file cpa.h.

5.68.2.3 reserved1

`Cpa32U _CpaPhysBufferList::reserved1`

Reserved for alignment

Definition at line 229 of file cpa.h.

5.68.2.4 flatBuffers

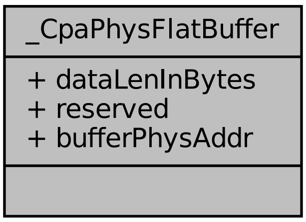
`CpaPhysFlatBuffer _CpaPhysBufferList::flatBuffers[ ]`

Array of flat buffer structures, of size numBuffers

Definition at line 231 of file cpa.h.

5.69 _CpaPhysFlatBuffer Struct Reference

Collaboration diagram for _CpaPhysFlatBuffer:



Data Fields

- Cpa32U dataLenInBytes
- Cpa32U reserved
- CpaPhysicalAddr bufferPhysAddr

5.69.1 Detailed Description

Flat buffer structure with physical address.

Description:

Functions taking this structure do not need to do any virtual to physical address translation before writing the buffer to hardware.

Definition at line 192 of file cpa.h.

5.69.2 Field Documentation

5.69.2.1 dataLenInBytes

`Cpa32U _CpaPhysFlatBuffer::dataLenInBytes`

Data length specified in bytes. When used as an input parameter to a function, the length specifies the current length of the buffer. When used as an output parameter to a function, the length passed in specifies the maximum length of the buffer on return (i.e. the allocated length). The implementation will not write past this length. On return, the length is always unchanged.

Definition at line 193 of file cpa.h.

5.69.2.2 reserved

`Cpa32U _CpaPhysFlatBuffer::reserved`

Reserved for alignment

Definition at line 202 of file cpa.h.

5.69.2.3 bufferPhysAddr

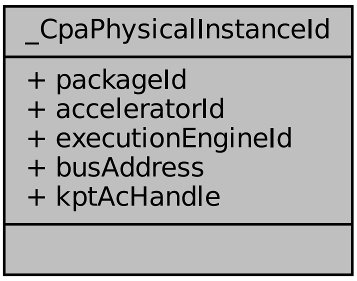
`CpaPhysicalAddr _CpaPhysFlatBuffer::bufferPhysAddr`

The physical address at which the data resides. The data pointed to is required to be in contiguous physical memory.

Definition at line 204 of file cpa.h.

5.70 _CpaPhysicalInstanceId Struct Reference

Collaboration diagram for _CpaPhysicalInstanceId:



Data Fields

- Cpa16U packageId
- Cpa16U acceleratorId
- Cpa16U executionEngineId
- Cpa16U busAddress
- Cpa32U kptAcHandle

5.70.1 Detailed Description

Physical Instance ID

Description:

Identifies the physical instance of an accelerator execution engine.

Accelerators grouped into "packages". Each accelerator can in turn contain one or more execution engines. Implementations of this API will define the packageId, acceleratorId, executionEngineId and busAddress as appropriate for the implementation. For example, for hardware-based accelerators, the packageId might identify the chip, which might contain multiple accelerators, each of which might contain multiple execution engines. The combination of packageId, acceleratorId and executionEngineId uniquely identifies the instance.

Hardware based accelerators implementing this API may also provide information on the location of the accelerator in the busAddress field. This field will be defined as appropriate for the implementation. For example, for PCIe attached accelerators, the busAddress may contain the PCIe bus, device and function number of the accelerators.

Definition at line 501 of file cpa.h.

5.70.2 Field Documentation

5.70.2.1 packageId

`Cpa16U _CpaPhysicalInstanceId::packageId`

Identifies the package within which the accelerator is contained.

Definition at line 502 of file cpa.h.

5.70.2.2 acceleratorId

`Cpa16U _CpaPhysicalInstanceId::acceleratorId`

Identifies the specific accelerator within the package.

Definition at line 505 of file cpa.h.

5.70.2.3 executionEngineId

`Cpa16U _CpaPhysicalInstanceId::executionEngineId`

Identifies the specific execution engine within the accelerator.

Definition at line 507 of file cpa.h.

5.70.2.4 busAddress

`Cpa16U _CpaPhysicalInstanceId::busAddress`

Identifies the bus address associated with the accelerator execution engine.

Definition at line 510 of file cpa.h.

5.70.2.5 kptAcHandle

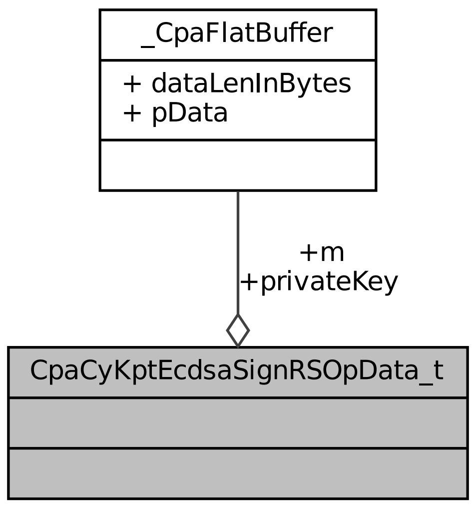
`Cpa32U _CpaPhysicalInstanceId::kptAcHandle`

Identifies the achandle of the accelerator.

Definition at line 513 of file cpa.h.

5.71 CpaCyKptEcdsaSignRSOpData_t Struct Reference

Collaboration diagram for CpaCyKptEcdsaSignRSOpData_t:



Data Fields

- CpaFlatBuffer privateKey
- CpaFlatBuffer m

5.71.1 Detailed Description

File: cpa_cy_kpt.h

KPT ECDSA Sign R & S Operation Data.

Description:

This structure contains the operation data for the cpaCyKptEcdsaSignRS function. The client MUST allocate the memory for this structure and the items pointed to by this structure. When the structure is passed into the function, ownership of the memory passes to the function. Ownership of the memory returns to the client when this structure is returned in the callback function. This key structure is encrypted when passed into cpaCyKptEcdsaSignRS Encrypt - AES-256-GCM (Key, AAD, Input) "||" - denotes concatenation

Key = SWK AAD = DER(OID) Input = (d) Encrypt (SWK, AAD, (d)) Output (AuthTag, EncryptedECKey)

privatekey == EncryptedECKey || AuthTag

OID's that shall be supported by KPT implementation: Curve OID DER(OID) secp256r1 1.2.840.10045.3.1.7 06 08 2A 86 48 CE 3D 03 01 07 secp384r1 1.3.132.0.34 06 05 2B 81 04 00 22 secp521r1 1.3.132.0.35 06 05 2B 81 04 00 23

Expected private key (d) sizes: secp256r1 256 bits secp384r1 384 bits secp521r1 576 bits (rounded up to a multiple of 64-bit quadword)

AuthTag is 128 bits (16 bytes)

For optimal performance all data buffers SHOULD be 8-byte aligned.

Note

If the client modifies or frees the memory referenced in this structure after it has been submitted to the cpaCyKptEcdsaSignRS function, and before it has been returned in the callback, undefined behavior will result.

See also

cpaCyEcdsaSignRS()

Definition at line 418 of file cpa_cy_kpt.h.

5.71.2 Field Documentation

5.71.2.1 privateKey

`CpaFlatBuffer CpaCyKptEcdsaSignRSOpData_t::privateKey`

Encrypted private key data of the form EncryptECKey || AuthTag

Definition at line 420 of file cpa_cy_kpt.h.

5.71.2.2 m

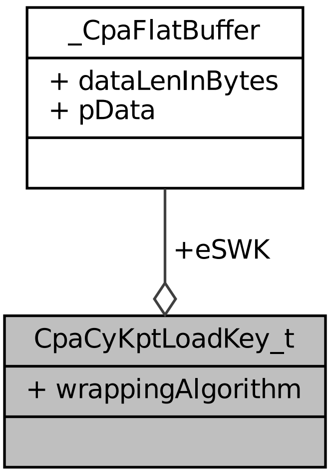
`CpaFlatBuffer CpaCyKptEcdsaSignRSOpData_t::m`

digest of the message to be signed

Definition at line 423 of file cpa_cy_kpt.h.

5.72 CpaCyKptLoadKey_t Struct Reference

Collaboration diagram for CpaCyKptLoadKey_t:



Data Fields

- CpaFlatBuffer eSWK
- CpaCyKptWrappingKeyType wrappingAlgorithm

5.72.1 Detailed Description

KPT Loading key format specification.

Description:

This structure defines the format of the symmetric wrapping key to be loaded into KPT. Application sets these parameters through the cpaCyKptLoadKey calls.

Definition at line 134 of file cpa_cy_kpt.h.

5.72.2 Field Documentation

5.72.2.1 eSWK

`CpaFlatBuffer CpaCyKptLoadKey_t::eSWK`

Encrypted SWK

Definition at line 136 of file cpa_cy_kpt.h.

5.72.2.2 wrappingAlgorithm

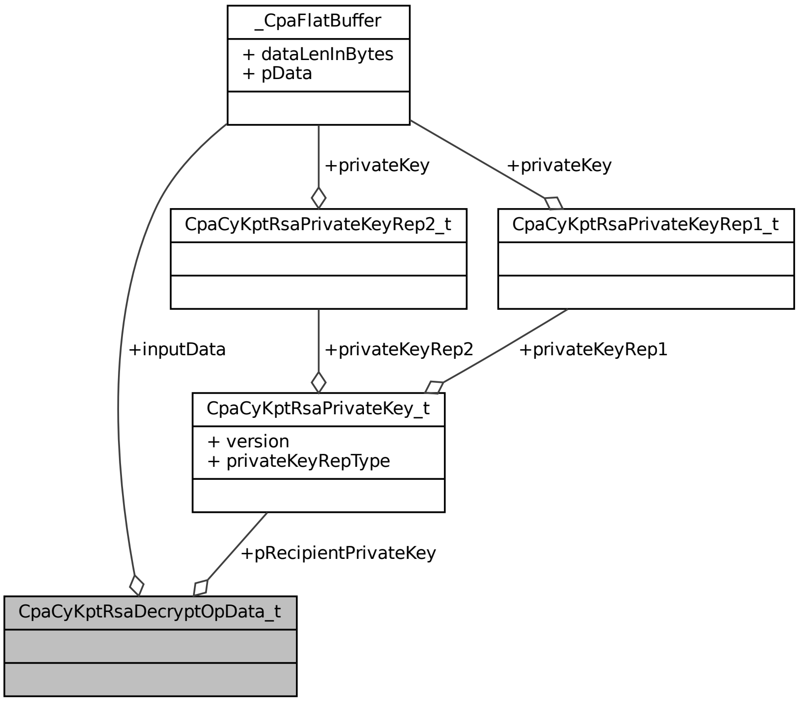
`CpaCyKptWrappingKeyType CpaCyKptLoadKey_t::wrappingAlgorithm`

Symmetric wrapping algorithm

Definition at line 138 of file cpa_cy_kpt.h.

5.73 CpaCyKptRsaDecryptOpData_t Struct Reference

Collaboration diagram for CpaCyKptRsaDecryptOpData_t:



Data Fields

- CpaCyKptRsaPrivateKey * pRecipientPrivateKey
- CpaFlatBuffer inputData

5.73.1 Detailed Description

File: cpa_cy_kpt.h

KPT RSA Decryption Primitive Operation Data

Description:

This structure lists the different items that are required in the cpaCyKptRsaDecrypt function. As the RSA decryption primitive and signature primitive operations are mathematically identical this structure may also be used to perform an RSA signature primitive operation. When performing an RSA decryption primitive operation, the input data is the cipher text and the output data is the message text. When performing an RSA signature primitive operation, the input data is the message and the output data is the signature. The client MUST allocate the memory for this structure. When the structure is passed into the function, ownership of the memory passes to he function. Ownership of the memory returns to the client when this structure is returned in the CpaCyGenFlatBufCbFunc callback function.

Note

If the client modifies or frees the memory referenced in this structure after it has been submitted to the cpaCyKptRsaDecrypt function, and before it has been returned in the callback, undefined behavior will result. All values in this structure are required to be in Most Significant Byte first order, e.g. inputData.pData[0] = MSB.

Definition at line 357 of file cpa_cy_kpt.h.

5.73.2 Field Documentation

5.73.2.1 pRecipientPrivateKey

`CpaCyKptRsaPrivateKey* CpaCyKptRsaDecryptOpData_t::pRecipientPrivateKey`

Pointer to the recipient's RSA private key.

Definition at line 359 of file cpa_cy_kpt.h.

5.73.2.2 inputData

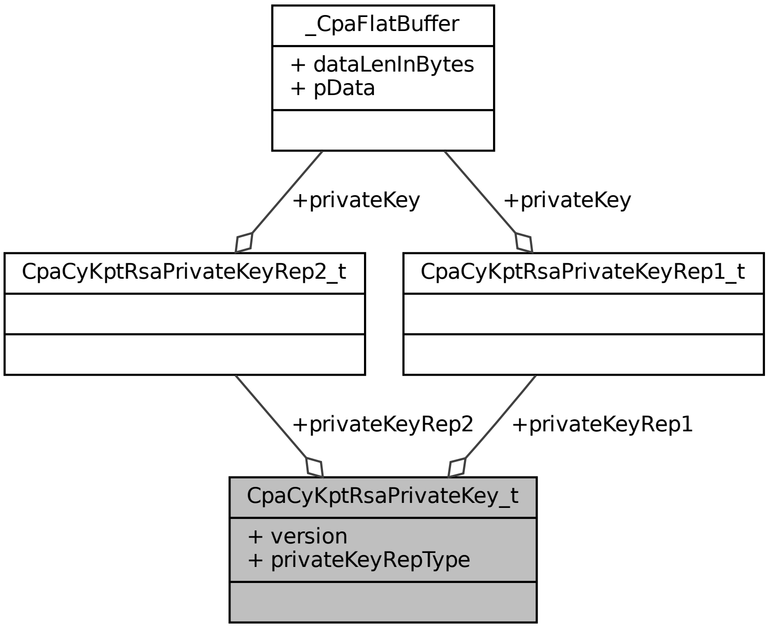
`CpaFlatBuffer CpaCyKptRsaDecryptOpData_t::inputData`

The input data that the RSA decryption primitive operation is performed on. The data pointed to is an integer that MUST be in big- endian order. The value MUST be between 0 and the modulus n - 1.

Definition at line 361 of file cpa_cy_kpt.h.

5.74 CpaCyKptRsaPrivateKey_t Struct Reference

Collaboration diagram for CpaCyKptRsaPrivateKey_t:



Data Fields

- CpaCyRsaVersion version
- CpaCyRsaPrivateKeyRepType privateKeyRepType
- CpaCyKptRsaPrivateKeyRep1 privateKeyRep1
- CpaCyKptRsaPrivateKeyRep2 privateKeyRep2

5.74.1 Detailed Description

File: cpa_cy_kpt.h

RSA Private Key Structure.

Description:

This structure contains the two representations that can be used for describing the RSA private key. The privateKeyRepType will be used to identify which representation is to be used. Typically, using the second representation results in faster decryption operations.

Definition at line 307 of file cpa_cy_kpt.h.

5.74.2 Field Documentation

5.74.2.1 version

`CpaCyRsaVersion CpaCyKptRsaPrivateKey_t::version`

Indicates the version of the PKCS #1 specification that is supported. Note that this applies to both representations.

Definition at line 309 of file cpa_cy_kpt.h.

5.74.2.2 privateKeyRepType

`CpaCyRsaPrivateKeyRepType CpaCyKptRsaPrivateKey_t::privateKeyRepType`

This value is used to identify which of the private key representation types in this structure is relevant. When performing key generation operations for Type 2 representations, memory must also be allocated for the type 1 representations, and values for both will be returned.

Definition at line 313 of file cpa_cy_kpt.h.

5.74.2.3 privateKeyRep1

`CpaCyKptRsaPrivateKeyRep1 CpaCyKptRsaPrivateKey_t::privateKeyRep1`

This is the first representation of the RSA private key as defined in the PKCS #1 V2.2 specification.

Definition at line 319 of file cpa_cy_kpt.h.

5.74.2.4 privateKeyRep2

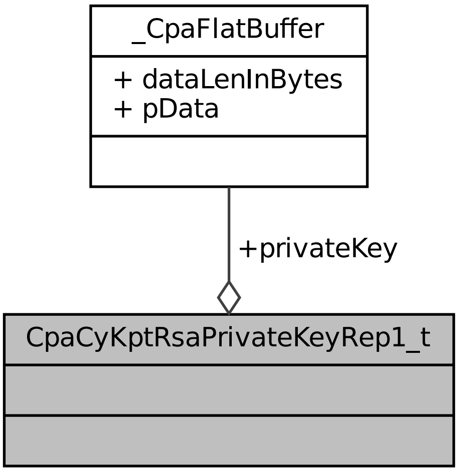
`CpaCyKptRsaPrivateKeyRep2 CpaCyKptRsaPrivateKey_t::privateKeyRep2`

This is the second representation of the RSA private key as defined in the PKCS #1 V2.2 specification.

Definition at line 322 of file cpa_cy_kpt.h.

5.75 CpaCyKptRsaPrivateKeyRep1_t Struct Reference

Collaboration diagram for CpaCyKptRsaPrivateKeyRep1_t:



Data Fields

- CpaFlatBuffer privateKey

5.75.1 Detailed Description

File: cpa_cy_kpt.h

RSA Private Key Structure For Representation 1.

Description:

> This structure contains the first representation that can be used for describing the RSA private key, represented by the tuple of the modulus (N) and the private exponent (D). The representation is encrypted as follows: Encrypt - AES-256-GCM (Key, AAD, Input) "||" - denotes concatenation Key = SWK AAD = DER(OID) Input = (D || N) Encrypt (SWK, AAD, (D || N)) Output (AuthTag, (D || N)') EncryptedRSAKey = (D || N)'

privateKey = (EncryptedRSAKey || AuthTag)

OID's that shall be supported by KPT implementation: OID DER(OID) 1.2.840.113549.1.1 06 08 2A 86 48 86 F7 0D 01 01

Permitted lengths for N and D are:

- 512 bits (64 bytes),
- 1024 bits (128 bytes),
- 1536 bits (192 bytes),
- 2048 bits (256 bytes),
- 3072 bits (384 bytes),
- 4096 bits (512 bytes), or
- 8192 bits (1024 bytes).

AuthTag is 128 bits (16 bytes)

Note

It is important that the value D is big enough. It is STRONGLY recommended that this value is at least half the length of the modulus N to protect against the Wiener attack.

Definition at line 234 of file cpa_cy_kpt.h.

5.75.2 Field Documentation

5.75.2.1 privateKey

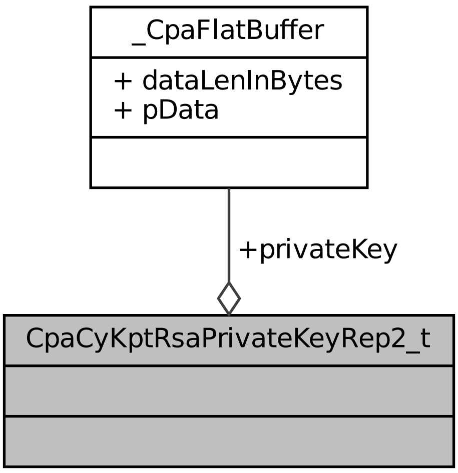
`CpaFlatBuffer CpaCyKptRsaPrivateKeyRep1_t::privateKey`

The EncryptedRSAKey concatenated with AuthTag

Definition at line 236 of file cpa_cy_kpt.h.

5.76 CpaCyKptRsaPrivateKeyRep2_t Struct Reference

Collaboration diagram for CpaCyKptRsaPrivateKeyRep2_t:



Data Fields

- CpaFlatBuffer privateKey

5.76.1 Detailed Description

File: cpa_cy_kpt.h

KPT RSA Private Key Structure For Representation 2.

Description:

This structure contains the second representation that can be used for describing the RSA private key. The quintuple of p, q, dP, dQ, and qInv (explained below and in the spec) are required for the second representation. For KPT the parameters are Encrypted with the assoicated SWK as follows: Encrypt - AES-256-GCM (Key, AAD, Input) "||" - denotes concatenation Key = SWK AAD = DER(OID) Input = (P || Q || dP || dQ || Qinv || publicExponentE) Expanded Description: Encrypt (SWK, AAD, (P || Q || dP || dQ || Qinv || publicExponentE)) EncryptedRSAKey = (P || Q || dP || dQ || Qinv || publicExponentE)' Output (AuthTag, EncryptedRSAKey)

privateKey = EncryptedRSAKey || AuthTag

OID's that shall be supported by KPT implementation: OID DER(OID) 1.2.840.113549.1.1 06 08 2A 86 48 86 F7 0D 01 01

All of the encrypted parameters will be of equal size. The length of each will be equal to keySize in bytes/2. For example for a key size of 256 Bytes (2048 bits), the length of P, Q, dP, dQ, and Qinv are all 128 Bytes, plus the publicExponentE of 256 Bytes, giving a total size for EncryptedRSAKey of 896 Bytes.

AuthTag is 128 bits (16 bytes)

Permitted Key Sizes are:

- 512 bits (64 bytes),
- 1024 bits (128 bytes),
- 1536 bits (192 bytes),
- 2048 bits (256 bytes),
- 3072 bits (384 bytes),
- 4096 bits (512 bytes), or
- 8192 bits (1024 bytes).

Definition at line 287 of file cpa_cy_kpt.h.

5.76.2 Field Documentation

5.76.2.1 privateKey

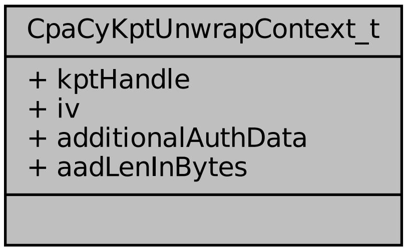
`CpaFlatBuffer CpaCyKptRsaPrivateKeyRep2_t::privateKey`

RSA private key representation 2 is built up from the tuple of p, q, dP, dQ, qInv, publicExponentE and AuthTag.

Definition at line 289 of file cpa_cy_kpt.h.

5.77 CpaCyKptUnwrapContext_t Struct Reference

Collaboration diagram for CpaCyKptUnwrapContext_t:



Data Fields

- CpaCyKptHandle kptHandle
- Cpa8U iv [CPA_CY_KPT_MAX_IV_LENGTH]
- Cpa8U additionalAuthData [CPA_CY_KPT_MAX_AAD_LENGTH]
- Cpa32U aadLenInBytes

5.77.1 Detailed Description

File: cpa_cy_kpt.h

Structure of KPT unwrapping context.

Description:

This structure is a parameter of KPT crypto APIs, it contains data relating to KPT WPK unwrapping, the application needs to fill in this information.

Definition at line 179 of file cpa_cy_kpt.h.

5.77.2 Field Documentation

5.77.2.1 kptHandle

`CpaCyKptHandle CpaCyKptUnwrapContext_t::kptHandle`

This is application's unique handle that identifies its (symmetric) wrapping key

Definition at line 181 of file cpa_cy_kpt.h.

5.77.2.2 iv

`Cpa8U CpaCyKptUnwrapContext_t::iv[CPA_CY_KPT_MAX_IV_LENGTH]`

Initialization Vector

Definition at line 184 of file cpa_cy_kpt.h.

5.77.2.3 additionalAuthData

`Cpa8U CpaCyKptUnwrapContext_t::additionalAuthData[CPA_CY_KPT_MAX_AAD_LENGTH]`

A buffer holding the Additional Authenticated Data.

Definition at line 186 of file cpa_cy_kpt.h.

5.77.2.4 aadLenInBytes

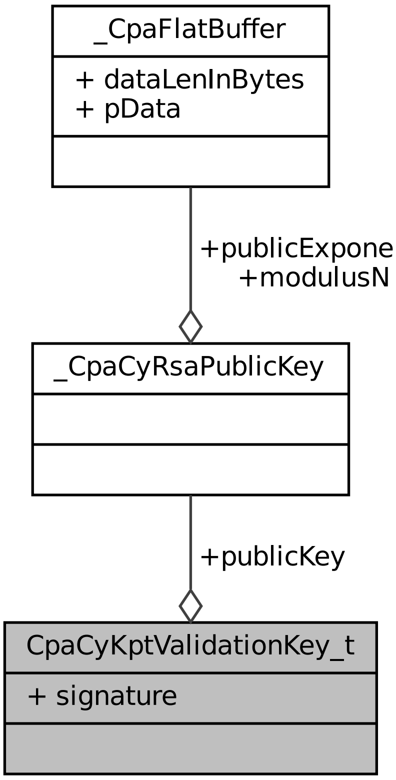
`Cpa32U CpaCyKptUnwrapContext_t::aadLenInBytes`

Number of bytes representing the size of AAD within additionalAuthData buffer.

Definition at line 188 of file cpa_cy_kpt.h.

5.78 CpaCyKptValidationKey_t Struct Reference

Collaboration diagram for CpaCyKptValidationKey_t:



Data Fields

- CpaCyRsaPublicKey publicKey
- Cpa8U signature [CPA_CY_RSA3K_SIG_SIZE_INBYTES]

5.78.1 Detailed Description

KPT device credentials key certificate

Description:

This structure defines the key format for use with KPT.

See also

cpaCyKptQueryDeviceCredentials

Definition at line 101 of file cpa_cy_kpt.h.

5.78.2 Field Documentation

5.78.2.1 publicKey

`CpaCyRsaPublicKey CpaCyKptValidationKey_t::publicKey`

Key

Definition at line 103 of file cpa_cy_kpt.h.

5.78.2.2 signature

`Cpa8U CpaCyKptValidationKey_t::signature[CPA_CY_RSA3K_SIG_SIZE_INBYTES]`

Signature of key

Definition at line 105 of file cpa_cy_kpt.h.