Side-Effects:

副作用:
None
没有人

Blocking:
阻止:
Yes when configured to operate in synchronous mode.
当配置为在同步模式下运行时，是。

Reentrant:
可重入:
No
不

Thread-safe:
线程安全:
Yes
是

Parameters:
参数:

`[in]` *instanceHandle* Instance handle.
[in] instanceHandle 执行个体控制代码。
`[in]` *pCb* Callback function pointer. If this is set to a NULL value the function will operate synchronously.
[in] pCb 回调函数指针。如果设置为空值，函数将同步运行。
`[in]` *pCallbackTag* User-supplied value to help identify request.
[in] pCallbackTag 使用者提供的值，可协助识别要求。
`[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.
[in] pOpData 结构，包含执行作业所需的所有资料。客户端代码为此结构分配内存。该组件取得内存的所有权，直到它在回调中被返回。
`[out]` *pProtocolStatus* The result passes/fails the DSA protocol related checks.
[out] pProtocolStatus 结果通过/未通过 DSA 协议相关检查。
`[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.
[out]DSA 参数 p、p 奇数和 2^(L-1 的 pP 候选)< p < X 在调用时，回调函数将在 pOut 参数中包含此参数。

Return values:

返回值:

CPA_STATUS_SUCCESS Function executed successfully.
CPA_STATUS_SUCCESS 函数执行成功。
CPA_STATUS_FAIL Function failed.
CPA_STATUS_FAIL 函数失败。
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.

CPA_STATUS_RETRY重新提交请求。传递的CPA_STATUS_INVALID_PARAM参数无效。CPA _状态_资源 与系统资源相关的错误。CPA _状态_重新启动 API 实现正在重新启动。重新提交请求。不支持CPA_STATUS_UNSUPPORTED函数。

Precondition:

前提条件:

The component has been initialized.
组件已初始化。

Postcondition:
后置条件:

None
没有人

Note:

注意:

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.

当 pCb 为非空时，会生成一个类型为 CpaCyDsaPParamGenCbFunc 的异步回调来响应此函数调用。为了获得最佳性能，数据指针应该 8 字节对齐。

See also:

另请参见:

CpaCyDsaPParamGenOpData, **CpaCyDsaGenCbFunc**

CpaCyDsaPParamGenOpData, CpaCyDsaGenCbFunc

```
cpaCyDsaGenGParam (  const                                                    instanceHandle, pCb
cpaCyDsaGenGParam (  const                                                    instanceHandle, pCb,
const
                                 void *pCallbackTag,
                                 const *  pOpData,
                                 *pProtocolStatus,
                                 *pG
                               )
```

13.7 Function Documentation

Generate DSA G Parameter.

生成DSA G参数。

This function performs FIPS 186-3 Appendix A.2.1, steps 1 and 3, and part of step 4:
该功能执行FIPS 186-3附录A.2.1的步骤1和3以及步骤4的一部分:

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.
2. e = (p - 1)/q. 3。设置g = h^e mod第4页。如果(g = 1)，则转到步骤2。这里，实现将检查g == 1，并相应地返回状态。

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.
在回调函数中作为参数protocol status(或者，在同步调用的情况下，在参数*pProtocolStatus中)返回的协议状态用于指示值g是否可接受。

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.
具体来说，(protocolStatus == CPA_TRUE)表示g是可接受的。同时，(protocolStatus == CPA_FALSE)意味着g == 1，因此应该使用不同的h值来生成另一个g值。

Context:
背景:

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.
当作为异步函数调用时，它不能休眠。它可以在不允许休眠的上下文中执行。当作为同步函数调用时，它可能会休眠。它不能在不允许休眠的上下文中执行。

Assumptions:
假设:

None
没有人

Side-Effects:
副作用:

None
没有人

Blocking:
阻止:

Yes when configured to operate in synchronous mode.
当配置为在同步模式下运行时，是。

Reentrant:
可重入:

No

不

Thread-safe:

线程安全:

Yes

是

Parameters:

参数:

[in] *instanceHandle* Instance handle.

[in] instanceHandle 执行个体控制代码。

[in] *pCb* Callback function pointer. If this is set to a NULL value the function will operate synchronously.

[in] pCb 回调函数指针。如果设置为空值，函数将同步运行。

[in] *pCallbackTag* User-supplied value to help identify request.

[in] pCallbackTag 使用者提供的值，可协助识别要求。

[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.

[in] pOpData 结构，包含执行作业所需的所有资料。客户端代码为此结构分配内存。该组件取得内存的所有权，直到它在回调中被返回。

[out] *pProtocolStatus* The result passes/fails the DSA protocol related checks.

[out] pProtocolStatus 结果通过/未通过 DSA 协议相关检查。

[out] *pG* g = h^((p-1)/q) mod p. On invocation the callback function will contain this parameter in the pOut parameter.

[out] pG g = h^((p-1)/q) mod p .在调用时，回调函数将在 pOut 参数中包含此参数。

Return values:

返回值:

CPA_STATUS_SUCCESS Function executed successfully.

CPA_STATUS_SUCCESS 函数执行成功。

CPA_STATUS_FAIL Function failed.

CPA_STATUS_FAIL 函数失败。

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.

CPA_STATUS_RETRY 重新提交请求。传递的 CPA_STATUS_INVALID_PARAM 参数无效。与系统资源相关的 CPA_STATUS_RESOURCE 错误。CPA_STATUS_RESTARTING API 实现正在重新启动。重新提交请求。不支持 CPA_STATUS_UNSUPPORTED 函数。

Precondition:

前提条件:

The component has been initialized via cpaCyStartInstance function.

该组件已通过 cpaCyStartInstance 函数初始化。

Postcondition:

后置条件:

None

没有人

Note:

注意:

W
h
e
n

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.
当pCb为非空时，会生成一个CpaCyDsaGParamGenCbFunc类型的异步回调来响应此函数调用。为了获得最佳性能，数据指针应该8字节对齐。

See also:

另请参见:

CpaCyDsaGParamGenOpData, **CpaCyDsaGenCbFunc**
CpaCyDsaGParamGenOpData, CpaCyDsaGenCbFunc

cpaCyDsaGenYParam (const instanceHandle, pCb, pCallbackTag,

```
cpaCyDsaGenYParam (  const                                  instanceHandle, pCb, pCallbackTag,
const
void *
                              const * pOpData,
                              *pProtocolStatus,
                              *pY
                             )
```

Generate DSA Y Parameter.
生成DSA Y参数。

This function performs modular exponentiation to generate y as described in FIPS 186-3 section 4.1: y = g^x mod p
此函数执行模幂运算以生成y，如FIPS 186-3第4.1节所述:y = gˆx模p

Context:
背景:

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.
当作为异步函数调用时，它不能休眠。它可以在不允许休眠的上下文中执行。当作为同步函数调用时，它可能会休眠。它不能在不允许休眠的上下文中执行。

Assumptions:
假设:

None
没有人

Side-Effects:
副作用:

None
没有人

Blocking:
阻止:

Yes when configured to operate in synchronous mode.
当配置为在同步模式下运行时，是。

Reentrant:

可重入:

No

不

Thread-safe:

线程安全:

Yes

是

Parameters:

参数:

`[in]` *instanceHandle* Instance handle.

[in] instanceHandle 执行个体控制代码。

`[in]` *pCb* Callback function pointer. If this is set to a NULL value the function will operate synchronously.

[in] pCb 回调函数指针。如果设置为空值，函数将同步运行。

`[in]` *pCallbackTag* User-supplied value to help identify request.

[in] pCallbackTag 使用者提供的值，可协助识别要求。

`[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.

[in] pOpData 结构，包含执行作业所需的所有资料。客户端代码为此结构分配内存。该组件取得内存的所有权，直到它在回调中被返回。

`[out]` *pProtocolStatus* The result passes/fails the DSA protocol related checks.

[out] pProtocolStatus 结果通过/未通过 DSA 协议相关检查。

`[out]` *pY* y = g^x mod p* On invocation the callback function will contain this parameter in the pOut parameter.

[out] pY y = gˆx mod p*在调用时，回调函数将在 pOut 参数中包含此参数。

Return values:

返回值:

CPA_STATUS_SUCCESS Function executed successfully.

CPA_STATUS_SUCCESS 函数执行成功。

CPA_STATUS_FAIL Function failed.

CPA_STATUS_FAIL 函数失败。

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.

CPA_STATUS_RETRY 重新提交请求。传递的 CPA_STATUS_INVALID_PARAM 参数无效。与系统资源相关的 CPA_STATUS_RESOURCE 错误。CPA_STATUS_RESTARTING API 实现正在重新启动。重新提交请求。不支持 CPA_STATUS_UNSUPPORTED 函数。

Precondition:

前提条件:

The component has been initialized via cpaCyStartInstance function.

该组件已通过 cpaCyStartInstance 函数初始化。

Postcondition:

后置条件:

None

没有人

Note:

注意:

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.
当pCb为非空时，会生成一个CpaCyDsaYParamGenCbFunc类型的异步回调来响应此函数调用。为了获得最佳性能，数据指针应该8字节对齐。

See also:

另请参见:

CpaCyDsaYParamGenOpData, **CpaCyDsaGenCbFunc**
CpaCyDsaYParamGenOpData, CpaCyDsaGenCbFunc

```
cpaCyDsaSignR (  const                                  instanceHandle, pCb
cpaCyDsaSignR (  const                                  instanceHandle, pCb,
const
                         void *pCallbackTag,
                         const * pOpData,
                         *pProtocolStatus,
                         *pR
                        )
```

Generate DSA R Signature.
生成DSA R签名。

This function generates the DSA R signature as described in FIPS 186-3 Section 4.6: r = (g^k mod p) mod q
此函数生成DSA R签名，如FIPS 186-3第4.6节所述: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.
在回调函数中作为参数protocol status(或者，在同步调用的情况下，在参数*pProtocolStatus中)返回的协议状态用于指示值r == 0。

Specifically, (protocolStatus == CPA_TRUE) means r != 0, while (protocolStatus == CPA_FALSE) means r
具体来说，(protocolStatus == CPA_TRUE)就是r! = 0，而(protocolStatus == CPA_FALSE)表示r
== 0.
== 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.
签名r的生成不依赖于正被签名的消息的内容，因此该操作可以针对不同的k值预先完成。然后，一旦每个消息变得可用，则只需要生成签名s。

Context:
背景:

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.
当作为异步函数调用时，它不能休眠。它可以在不允许休眠的上下文中执行。当作为同步函数调用时，它可能会休眠。它不能在不允许休眠的上下文中执行。

Assumptions:
假设:

None
没有人

Side-Effects:

副作用:
None
没有人

Blocking:
阻止:

Yes when configured to operate in synchronous mode.
当配置为在同步模式下运行时，是。

Reentrant:
可重入:

No
不

Thread-safe:
线程安全:

Yes
是

Parameters:
参数:

`[in]` *instanceHandle* Instance handle.
[in] instanceHandle 执行个体控制代码。
`[in]` *pCb* Callback function pointer. If this is set to a NULL value the function will operate synchronously.
[in] pCb 回调函数指针。如果设置为空值，函数将同步运行。
`[in]` *pCallbackTag* User-supplied value to help identify request.
[in] pCallbackTag 使用者提供的值，可协助识别要求。
`[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.
[in] pOpData 结构，包含执行作业所需的所有资料。客户端代码为此结构分配内存。该组件取得内存的所有权，直到它在回调中被返回。
`[out]` *pProtocolStatus* The result passes/fails the DSA protocol related checks.
[out] pProtocolStatus 结果通过/未通过 DSA 协议相关检查。
`[out]` *pR* DSA message signature r. On invocation the callback function will contain this parameter in the pOut parameter.
[out] pR DSA 消息签名 r .在调用回调函数时，该函数将在 pOut 参数中包含此参数。

Return values:

返回值:

CPA_STATUS_SUCCESS Function executed successfully.
CPA_STATUS_SUCCESS 函数执行成功。
CPA_STATUS_FAIL Function failed.
CPA_STATUS_FAIL 函数失败。
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.

CPA_STATUS_RETRY 重新提交请求。传递的 CPA_STATUS_INVALID_PARAM 参数无效。与系统资源相关的 CPA_STATUS_RESOURCE 错误。CPA_STATUS_RESTARTING API 实现正在重新启动。重新提交请求。不支持 CPA_STATUS_UNSUPPORTED 函数。

Precondition:

前提条件:

The component has been initialized via cpaCyStartInstance function.

该组件已通过 cpaCyStartInstance 函数初始化。

Postcondition:

后置条件:

None

没有人

Note:

注意:

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. 当 pCb 为非空时，会生成一个 CpaCyDsaRSignCbFunc 类型的异步回调来响应此函数调用。为了获得最佳性能，数据指针应该 8 字节对齐。

See also:

另请参见:

CpaCyDsaRSignOpData, **CpaCyDsaGenCbFunc**, **cpaCyDsaSignS()**, **cpaCyDsaSignRS()**

CpaCyDsaRSignOpData, CpaCyDsaGenCbFunccpaCyDsaSignS()cpaCyDsaSignRS()

```
cpaCyDsaSignS ( const                                instanceHandle, pCb
cpaCyDsaSignS (  const                               instanceHandle, pCb,
const
                      void *pCallbackTag,
                      const * pOpData,
                      *pProtocolStatus,
                      *pS
                    )
```

13.7 Function Documentation

Generate DSA S Signature.

生成DSA的签名。

This function generates the DSA S signature as described in FIPS 186-3 Section 4.6: s = (k^-1(z + xr)) mod q
该函数生成DSA的签名，如FIPS 186-3第4.6节所述: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.
这里，z = Hash(M)最左边的min(N，outlen)位。此函数不执行SHA摘要；z由调用者计算，并作为参数在pOpData字段中传递。

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.
在回调函数中作为参数protocol status(或者，在同步调用的情况下，在参数*pProtocolStatus中)返回的协议状态用于指示值s == 0。

Specifically, (protocolStatus == CPA_TRUE) means s != 0, while (protocolStatus == CPA_FALSE) means s
具体来说，(protocolStatus == CPA_TRUE)的意思是s！= 0，而(protocolStatus == CPA_FALSE)表示s
== 0.
== 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.
如果已经预先生成了签名r，那么一旦消息变得可用，就可以使用该函数来生成签名s。

Context:
背景:

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.
当作为异步函数调用时，它不能休眠。它可以在不允许休眠的上下文中执行。当作为同步函数调用时，它可能会休眠。它不能在不允许休眠的上下文中执行。

Assumptions:
假设:

None
没有人

Side-Effects:
副作用:

None
没有人

Blocking:
阻止:

Yes when configured to operate in synchronous mode.
当配置为在同步模式下运行时，是。

Reentrant:
可重入:

No
不

Thread-safe:
线程安全:

Yes
是

Parameters:
参数:

`[in]` *instanceHandle* Instance handle.
[in] instanceHandle 执行个体控制代码。

`[in]` *pCb* Callback function pointer. If this is set to a NULL value the function will operate synchronously.
[in] pCb 回调函数指针。如果设置为空值，函数将同步运行。

`[in]` *pCallbackTag* User-supplied value to help identify request.
[in] pCallbackTag 使用者提供的值，可协助识别要求。

`[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.
[in] pOpData 结构，包含执行作业所需的所有资料。客户端代码为此结构分配内存。该组件取得内存的所有权，直到它在回调中被返回。

`[out]` *pProtocolStatus* The result passes/fails the DSA protocol related checks.
[out] pProtocolStatus 结果通过/未通过 DSA 协议相关检查。

`[out]` *pS* DSA message signature s. On invocation the callback function will contain this parameter in the pOut parameter.
[out] pS DSA 消息签名，在调用回调函数时，pOut 参数中将包含此参数。

Return values:
返回值:

CPA_STATUS_SUCCESS Function executed successfully.
CPA_STATUS_SUCCESS 函数执行成功。

CPA_STATUS_FAIL Function failed.
CPA_STATUS_FAIL 函数失败。

CPA_STATUS_RETRY Resubmit the request.
CPA_STATUS_INVALID_PARAM Invalid parameter passed in.
CPA_STATUS_RESOURCE Error related to system resources.

CPA_STATUS_RETRY 重新提交请求。传递的 CPA_STATUS_INVALID_PARAM 参数无效。与系统资源相关的 CPA_STATUS_RESOURCE 错误。

CPA_STATUS_RESTARTING API implementation is restarting. Resubmit the request.

CPA_STATUS_RESTARTING API 实现正在重新启动。重新提交请求。

CPA_STATUS_UNSUPPORTED Function is not supported.

不支持 CPA_STATUS_UNSUPPORTED 函数。

Precondition:

前提条件:

The component has been initialized via cpaCyStartInstance function.

该组件已通过 cpaCyStartInstance 函数初始化。

Postcondition:

后置条件:

None

没有人

Note:

注意:

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. 当pCb为非空时，会生成一个CpaCyDsaSSignCbFunc类型的异步回调来响应此函数调用。为了获得最佳性能，数据指针应该8字节对齐。

See also:

另请参见:

CpaCyDsaSSignOpData, **CpaCyDsaGenCbFunc**, **cpaCyDsaSignR()**, **cpaCyDsaSignRS()**

CpaCyDsaSSignOpData,CpaCyDsaGenCbFunccpaCyDsaSignR()cpaCyDsaSignRS()

cpaCyDsaSignRS (const instanceHandle

```
cpaCyDsaSignRS (  const                                    instanceHandle,
                       const pCb,
                       void *pCallbackTag,
                       const * pOpData,
                       *pProtocolStatus,
                       *pR,
                       *pS
                      )
```

Generate DSA R and S Signatures.
生成DSA R和S签名。

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

此函数生成DSA R和s签名，如FIPS 186-3第4.6节所述:r =(g^k mod p)modq s =(k^-

1(z+xr))modq

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.
这里，z = Hash(M)最左边的min(N，outlen)位。此函数不执行SHA摘要；z由调用者计算，并作为参数在pOpData字段中传递。

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.
在回调函数中作为参数protocol status(或者，在同步调用的情况下，在参数*pProtocolStatus中)返回的协议状态用于指示值r或s是否为零。

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)).
具体来说，(protocolStatus == CPA_TRUE)意味着两者都不为零(即(r! = 0) && (s! = 0))，而(protocolStatus == CPA_FALSE)表示r或s中至少有一个为零(即(r == 0) || (s == 0))。

Context:
背景:

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.

当作为异步函数调用时，它不能休眠。它可以在不允许休眠的上下文中执行。当作为同步函数调用时，它可能会休眠。它不能在不允许休眠的上下文中执行。

Assumptions:
假设:

None
没有人

Side-Effects:
副作用:

None
没有人

Blocking:
阻止:

Yes when configured to operate in synchronous mode.
当配置为在同步模式下运行时，是。

Reentrant:
可重入:

No
不

Thread-safe:

线程安全:

Yes
是

Parameters:
参数:

`[in]` *instanceHandle* Instance handle.
[in] instanceHandle 执行个体控制代码。

`[in]` *pCb* Callback function pointer. If this is set to a NULL value the function will operate synchronously.
[in] pCb 回调函数指针。如果设置为空值，函数将同步运行。

`[in]` *pCallbackTag* User-supplied value to help identify request.
[in] pCallbackTag 使用者提供的值，可协助识别要求。

`[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.
[in] pOpData 结构，包含执行作业所需的所有资料。客户端代码为此结构分配内存。该组件取得内存的所有权，直到它在回调中被返回。

`[out]` *pProtocolStatus* The result passes/fails the DSA protocol related checks.
[out] pProtocolStatus 结果通过/未通过 DSA 协议相关检查。

`[out]` *pR* DSA message signature r.
[out] pR DSA 消息签名。

`[out]` *pS* DSA message signature s.
[out] pS DSA 消息签名。

Return values:
返回值:

CPA_STATUS_SUCCESS Function executed successfully.
CPA_STATUS_SUCCESS 函数执行成功。

CPA_STATUS_FAIL Function failed.
CPA_STATUS_FAIL 函数失败。

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.

CPA_STATUS_RETRY 重新提交请求。传递的 CPA_STATUS_INVALID_PARAM 参数无效。与系统资源相关的 CPA_STATUS_RESOURCE 错误。CPA_STATUS_RESTARTING API 实现正在重新启动。重新提交请求。不支持 CPA_STATUS_UNSUPPORTED 函数。

Precondition:

前提条件:

The component has been initialized via cpaCyStartInstance function.
该组件已通过 cpaCyStartInstance 函数初始化。

Postcondition:
后置条件:

None
没有人

13.7 Function Documentation

Note:

注意:

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. 当pCb为非空时，会生成一个CpaCyDsaRSSignCbFunc类型的异步回调来响应此函数调用。为了获得最佳性能，数据指针应该8字节对齐。

See also:

另请参见:

CpaCyDsaRSSignOpData, **CpaCyDsaRSSignCbFunc**, **cpaCyDsaSignR()**, **cpaCyDsaSignS()**
CpaCyDsaRSSignOpData,CpaCyDsaRSSignCbFunccpaCyDsaSignR()cpaCyDsaSignS()

```
cpaCyDsaVerify ( const                                    instanceHandle
cpaCyDsaVerify  (  const                                  instanceHandle,
                        const pCb,
                        void *pCallbackTag,
                        const * pOpData,
                        *pVerifyStatus
                       )
```

Verify DSA R and S signatures.
验证DSA R和S签名。

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
该函数执行FIPS 186-3第4.7节:w =(s')ˆ-1 mod q u1 =(zw)mod q U2 =((r')w)mod q v =(((g)ˆu1(y)ˆu2)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.
这里，z = Hash(M')的最左边的min(N，outlen)位。此函数不执行SHA摘要；z由调用者计算，并作为参数在pOpData字段中传递。

A response status of ok (verifyStatus == CPA_TRUE) means v = r'. A response status of not ok (verifyStatus == CPA_FALSE) means v != r'.
ok响应状态(verifyStatus == CPA_TRUE)表示v = r'。响应状态不正常(verifyStatus == CPA_FALSE)意味着v！= r '。

Context:

背景:

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.

当作为异步函数调用时，它不能休眠。它可以在不允许休眠的上下文中执行。当作为同步函数调用时，它可能会休眠。它不能在不允许休眠的上下文中执行。

Assumptions:
假设:

None
没有人

Side-Effects:
副作用:

None
没有人

Blocking:
阻止:

Yes when configured to operate in synchronous mode.
当配置为在同步模式下运行时，是。

Reentrant:
可重入:

No
不

Thread-safe:
线程安全:

Yes
是

Parameters:
参数:

`[in]` *instanceHandle* Instance handle.
[in] instanceHandle 执行个体控制代码。
`[in]` *pCb* Callback function pointer. If this is set to a NULL value the function will operate synchronously.
[in] pCb 回调函数指针。如果设置为空值，函数将同步运行。
`[in]` *pCallbackTag* User-supplied value to help identify request.
[in] pCallbackTag 使用者提供的值，可协助识别要求。
`[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.
[in] pOpData 结构，包含执行作业所需的所有资料。客户端代码为此结构分配内存。该组件取得内存的所有权，直到它在回调中被返回。
`[out]` *pVerifyStatus* The verification passed or failed.
[out] pVerifyStatus 验证通过或失败。

Return values:

返回值：

CPA_STATUS_SUCCESS Function executed successfully.

CPA_STATUS_SUCCESS 函数执行成功。

CPA_STATUS_FAIL Function failed.

CPA_STATUS_FAIL 函数失败。

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.

CPA_STATUS_RETRY 重新提交请求。传递的 CPA_STATUS_INVALID_PARAM 参数无效。与系统资源相关的 CPA_STATUS_RESOURCE 错误。CPA_STATUS_RESTARTING API 实现正在重新启动。重新提交请求。不支持 CPA_STATUS_UNSUPPORTED 函数。

Precondition:

前提条件：

The component has been initialized via cpaCyStartInstance function.

该组件已通过 cpaCyStartInstance 函数初始化。

Postcondition:

后置条件：

None

没有人

Note:

注意：

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.

当 pCb 为非空时，会生成一个 CpaCyDsaVerifyCbFunc 类型的异步回调来响应此函数调用。为了获得最佳性能，数据指针应该 8 字节对齐。

See also:

另请参见:

CpaCyDsaVerifyOpData, **CpaCyDsaVerifyCbFunc**

CpaCyDsaVerifyOpData, CpaCyDsaVerifyCbFunc

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

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

Query statistics for a specific DSA instance.

查询特定DSA实例的统计信息。

Deprecated:

Deprecated:

As of v1.3 of the Crypto API, this function has been deprecated, replaced by
从Crypto API 1.3版开始，此函数已被弃用，由
cpaCyDsaQueryStats64().
cpaCyDsaQueryStats64()。

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.
该函数将查询DSA实现的特定实例的统计信息。用户必须分配CpaCyDsaStats结构，并将对该结构的引用传递到此函数调用中。该函数将统计结果写入传入的CpaCyDsaStats结构中。

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.
注意:此函数返回的统计数据不会中断当前的数据处理，因此可能会与统计数据检索过程中正在进行的操作稍微不同步。

Context:

背景:

This is a synchronous function and it can sleep. It MUST NOT be executed in a context that DOES NOT permit sleeping.
这是一个同步功能，它可以休眠。它不能在不允许休眠的上下文中执行。

Assumptions:

假设:

None
没有人

Side-Effects:

副作用:

None
没有人

Blocking:

阻止:

This function is synchronous and blocking.
这个函数是同步的和阻塞的。

Reentrant:

可重入:

No
不

Thread-safe:

线程安全:

Yes

是

Parameters:

参数:

`[in]` *instanceHandle* Instance handle.

[in] instanceHandle 执行个体控制代码。

`[out]` *pDsaStats* Pointer to memory into which the statistics will be written.

[out]PDS stats 指向将写入统计信息的内存的指针。

Return values:

返回值:

CPA_STATUS_SUCCESS Function executed successfully.

CPA_STATUS_SUCCESS 函数执行成功。

CPA_STATUS_FAIL Function failed.

CPA_STATUS_INVALID_PARAM Invalid parameter passed in.

CPA_STATUS_RESOURCE Error related to system resources.

CPA_STATUS_FAIL 函数失败。传递的 CPA_STATUS_INVALID_PARAM 参数无效。与系统资源相关的 CPA_STATUS_RESOURCE 错误。

CPA_STATUS_RESTARTING API implementation is restarting. Resubmit the request.

CPA_STATUS_RESTARTING API 实现正在重新启动。重新提交请求。

CPA_STATUS_UNSUPPORTED Function is not supported.

不支持 CPA_STATUS_UNSUPPORTED 函数。

Precondition:

前提条件:

Component has been initialized.

组件已初始化。

Postcondition:

后置条件:

None

没有人

Note:

注意:

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

该函数以同步方式运行，不会生成异步回调。

See also:

另请参见:

CpaCyDsaStats

CpaCyDsaStats

CpaStatus cpaCyDsaQueryStats64 (const **CpaInstanceHandle** *instanceHandle*,
CpaCyDsaStats64 * *pDsaStats*
ˋ

CpaStatus cpaCyDsaQueryStats64 (const **CpaInstanceHandle** *instanceHandle*,
CpaCyDsaStats64 * *pDsaStats*
ˋ

Query 64-bit statistics for a specific DSA instance.
查询特定DSA实例的64位统计信息。

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.
该函数将查询DSA实现的特定实例，以获得64位统计信息。用户必须分配CpaCyDsaStats64结构，并将对该结构的引用传递给此函数。该函数将统计结果写入传入的CpaCyDsaStats64结构中。

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.
注意:此函数返回的统计数据不会中断当前的数据处理，因此可能会与统计数据检索过程中正在进行的操作稍微不同步。

Context:
背景:

This is a synchronous function and it can sleep. It MUST NOT be executed in a context that DOES NOT permit sleeping.
这是一个同步功能，它可以休眠。它不能在不允许休眠的上下文中执行。

Assumptions:
假设:

None
没有人

Side-Effects:
副作用:

None
没有人

Blocking:
阻止:

This function is synchronous and blocking.
这个函数是同步的和阻塞的。

Reentrant:
可重入:

No
不

Thread-safe:
线程安全:

Yes
是

Parameters:
参数:

`[in]` *instanceHandle* Instance handle.
[in] instanceHandle 执行个体控制代码。
`[out]` *pDsaStats* Pointer to memory into which the statistics will be written.
[out]PDS stats 指向将写入统计信息的内存的指针。

Return values:
返回值:

CPA_STATUS_SUCCESS Function executed successfully.
CPA_STATUS_SUCCESS 函数执行成功。
CPA_STATUS_FAIL Function failed.
CPA_STATUS_INVALID_PARAM Invalid parameter passed in.
CPA_STATUS_RESOURCE Error related to system resources.
CPA_STATUS_FAIL 函数失败。传递的 CPA_STATUS_INVALID_PARAM 参数无效。与系统资源相关的 CPA_STATUS_RESOURCE 错误。
CPA_STATUS_RESTARTING API implementation is restarting. Resubmit the request.
CPA_STATUS_RESTARTING API 实现正在重新启动。重新提交请求。
CPA_STATUS_UNSUPPORTED Function is not supported.
不支持 CPA_STATUS_UNSUPPORTED 函数。

Precondition:
前提条件:

Component has been initialized.
组件已初始化。

Postcondition:
后置条件:

None
没有人

Note:

注意:

This function operates in a synchronous manner and no asynchronous callback will be generated.
该函数以同步方式运行，不会生成异步回调。

See also:

另请参见:

CpaCyDsaStats
CpaCyDsaStats

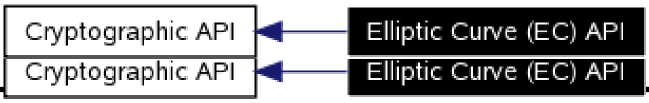
14 Elliptic Curve (EC) API

15 椭圆曲线(EC) API

[Cryptographic API]

[Cryptographic API]

Collaboration diagram for Elliptic Curve (EC) API:
椭圆曲线(EC) API 的协作图:



15.1 Detailed Description

15.2 详细描述

File: cpa_cy_ec.h

文件:cpa_cy_ec.h

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

All implementations will support at least the following:

这些函数指定了公钥加密(加密)椭圆曲线(EC)操作的 API。所有实施将至少支持以下内容:

- "NIST RECOMMENDED ELLIPTIC CURVES FOR FEDERAL GOVERNMENT USE" as defined by
- “NIST 建议联邦政府使用椭圆曲线”, 定义如下
  http://csrc.nist.gov/groups/ST/toolkit/documents/dss/NISTReCur.pdf
  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
- max(log2(q), log2(n) + log2(h)) <= 512 的随机曲线，其中 q 是模数，n 是曲线的阶，h 是余因子

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
对于 Montgomery 和 Edwards 25519 和 448 椭圆曲线，支持以下操作:1 .蒙哥马利 25519 曲线|标量点乘输入:P 点的蒙哥马利仿射坐标 X 标量 k 输出:P 点的蒙哥马利仿射坐标 X[k/P 解码:标量 k 总是由实现解码

3. 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
4. 蒙哥马利25519曲线|生成器点乘输入:标量k输出:点[k]G的蒙哥马利仿射坐标X解码:标量k始终由实现解码

5. 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 specify if decoding is required
6. 扭曲的Edwards 25519曲线|标量点乘法输入:点P的扭曲的Edwards仿射坐标X扭曲的Edwards仿射坐标Y点P的标量k输出:点[k]P的扭曲的Edwards仿射坐标X点[k]P的扭曲的Edwards仿射坐标Y解码:调用方必须指定是否需要解码

7. 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 specify if decoding is required
8. 扭曲的Edwards 25519曲线|生成器点乘法输入:标量k输出:扭曲的Edwards仿射坐标X点[k]G扭曲的Edwards仿射坐标Y点[k]G解码:调用方必须指定是否需要解码

9. 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
10. 蒙哥马利448曲线|标量点乘输入:点P的蒙哥马利仿射坐标X标量k输出:点[k]的蒙哥马利仿射坐标X P解码:标量k始终由实现解码

11. 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
12. 蒙哥马利448曲线|生成器点乘输入:标量k输出:点[k]G的蒙哥马利仿射坐标X解码:标量k始终由实现解码

13. 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 specify if decoding is required
14. Edwards 448曲线|标量点乘输入:点P的Edwards仿射坐标X点P的Edwards仿射坐标Y标量k输出:点[k]的Edwards仿射坐标X点[k]P的Edwards仿射坐标Y点[k]P解码:调用方必须指定是否需要解码

15. 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 specify if decoding is required
16. Edwards 448曲线|生成器点乘法输入:标量k输出:点[k]的Edwards仿射坐标X G点[k]的Edwards仿射坐标Y解码:调用方必须指定是否需要解码

Note:
注意:

14.1 Detailed Description

14.2 详细描述

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

大数在 QuickAssist API 上表示，如大数 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.
此外，对于椭圆曲线运算，传递给 API 的大数的位长度不得超过 576 位。

14.3 Data Structures

14.4 数据结构

- struct **_CpaCyEcPointMultiplyOpData**

- 结构体_CpaCyEcPointMultiplyOpData
- struct **_CpaCyEcPointVerifyOpData**
- 结构体_CpaCyEcPointVerifyOpData
- struct **_CpaCyEcMontEdwdsPointMultiplyOpData**
- 结构体_CpaCyEcMontEdwdsPointMultiplyOpData
- struct **_CpaCyEcStats64**
- 结构体_CpaCyEcStats64

14.5 Typedefs

14.6 类型定义

- typedef enum **_CpaCyEcFieldType CpaCyEcFieldType**

- typedef 枚举_CpaCyEcFieldType CpaCyEcFieldType
- typedef enum **_CpaCyEcMontEdwdsCurveType CpaCyEcMontEdwdsCurveType**
- typedef 枚举_CpaCyEcMontEdwdsCurveType CpaCyEcMontEdwdsCurveType
- typedef **_CpaCyEcPointMultiplyOpData CpaCyEcPointMultiplyOpData**
- 数据类型说明_CpaCyEcPointMultiplyOpData CpaCyEcPointMultiplyOpData
- typedef **_CpaCyEcPointVerifyOpData CpaCyEcPointVerifyOpData**
- 数据类型说明_CpaCyEcPointVerifyOpData CpaCyEcPointVerifyOpData
- typedef **_CpaCyEcMontEdwdsPointMultiplyOpData CpaCyEcMontEdwdsPointMultiplyOpData**
- 数据类型说明_CpaCyEcMontEdwdsPointMultiplyOpData CpaCyEcMontEdwdsPointMultiplyOpData
- typedef **_CpaCyEcStats64 CpaCyEcStats64**
- 数据类型说明_CpaCyEcStats64 CpaCyEcStats64
- typedef void(* **CpaCyEcPointMultiplyCbFunc**)(void *pCallbackTag, **CpaStatus** status, void
- typedef void(*CpaCyEcPointMultiplyCbFunc CpaStatus
  *pOpData, **CpaBoolean** multiplyStatus, **CpaFlatBuffer** *pXk, **CpaFlatBuffer** *pYk)
  *pOpData，CpaBoolean CpaFlatBuffer CpaFlatBuffer
- typedef void(* **CpaCyEcPointVerifyCbFunc**)(void *pCallbackTag, **CpaStatus** status, void
- typedef void(*CpaCyEcPointVerifyCbFunc CpaStatus
  *pOpData, **CpaBoolean** verifyStatus)
  *pOpData，CpaBoolean

14.7 Enumerations

14.8 列举

- enum **_CpaCyEcFieldType** {
  CPA_CY_EC_FIELD_TYPE_PRIME,
  CPA_CY_EC_FIELD_TYPE_BINARY

- 列举型别_CpaCyEcFieldType
  CPA_CY_EC_FIELD_TYPE_PRIMECPA_CY_E
  C_FIELD_TYPE_BINARY
  }
  }
- enum **_CpaCyEcMontEdwdsCurveType** {
  CPA_CY_EC_MONTEDWDS_CURVE25519_TYPE,
  CPA_CY_EC_MONTEDWDS_ED25519_TYPE,
  CPA_CY_EC_MONTEDWDS_CURVE448_TYPE,
  CPA_CY_EC_MONTEDWDS_ED448_TYPE
- 列举型别_CpaCyEcMontEdwdsCurveType
  CPA_CY_EC_MONTEDWDS_CURVE25519_TYPECPA_CY_EC_MO
  NTEDWDS_ED25519_TYPECPA_CY_EC_MONTEDWDS_CURVE44
  8_TYPECPA_CY_EC_MONTEDWDS_ED448_TYPE
  }
  }

14.9 Functions

14.10 功能

- **CpaStatus cpaCyEcPointMultiply** (const **CpaInstanceHandle** instanceHandle, const

- CpaStatus cpaCyEcPointMultiply (常量 CpaInstanceHandle
  CpaCyEcPointMultiplyCbFunc pCb, void *pCallbackTag, const **CpaCyEcPointMultiplyOpData**
  CpaCyEcPointMultiplyCbFunc pCb, void *pCallbackTag, constCpaCyEcPointMultiplyOpData
  *pOpData, **CpaBoolean** *pMultiplyStatus, **CpaFlatBuffer** *pXk, **CpaFlatBuffer** *pYk)
  *pOpData, CpaBoolean CpaFlatBuffer CpaFlatBuffer
- **CpaStatus cpaCyEcPointVerify** (const **CpaInstanceHandle** instanceHandle, const
- CpaStatus cpaCyEcPointVerify (常量 CpaInstanceHandle
  CpaCyEcPointVerifyCbFunc pCb, void *pCallbackTag, const **CpaCyEcPointVerifyOpData**
  CpaCyEcPointVerifyCbFunc pCb, void *pCallbackTag, constCpaCyEcPointVerifyOpData
  *pOpData, **CpaBoolean** *pVerifyStatus)
  *pOpData, CpaBoolean
- **CpaStatus cpaCyEcMontEdwdsPointMultiply** (const **CpaInstanceHandle** instanceHandle, const **CpaCyEcPointMultiplyCbFunc** pCb, void *pCallbackTag, const **CpaCyEcMontEdwdsPointMultiplyOpData** *pOpData, **CpaBoolean** *pMultiplyStatus, **CpaFlatBuffer** *pXk, **CpaFlatBuffer** *pYk)
- CpaStatus cpaCyEcMontEdwdsPointMultiply (常量 CpaInstanceHandle
  CpaCyEcPointMultiplyCbFunc CpaCyEcMontEdwdsPointMultiplyOpData CpaBoolean CpaFlatBuffer
  CpaFlatBuffer
- **CpaStatus cpaCyEcQueryStats64** (const **CpaInstanceHandle** instanceHandle, **CpaCyEcStats64**
- CpaStatus cpaCyEcQueryStats64 (常量 CpaInstanceHandle CpaCyEcStats64
  *pEcStats)
  *pEcStats)
- **CpaStatus cpaCyKptEcPointMultiply** (const **CpaInstanceHandle** instanceHandle, const
- CpaStatus cpaCyKptEcPointMultiply (常量 CpaInstanceHandle

14.5 Functions

14.6 功能

CpaCyEcPointMultiplyCbFunc pCb, void *pCallbackTag, const **CpaCyEcPointMultiplyOpData**

CpaCyEcPointMultiplyCbFunc pCb，void *pCallbackTag，constCpaCyEcPointMultiplyOpData
*pOpData, **CpaBoolean** *pMultiplyStatus, **CpaFlatBuffer** *pXk, **CpaFlatBuffer** *pYk, **CpaFlatBuffer**
*pOpData，CpaBoolean CpaFlatBuffer CpaFlatBuffer CpaFlatBuffer
*pKptUnwrapContext)
*pKptUnwrapContext)

14.7 Data Structure Documentation

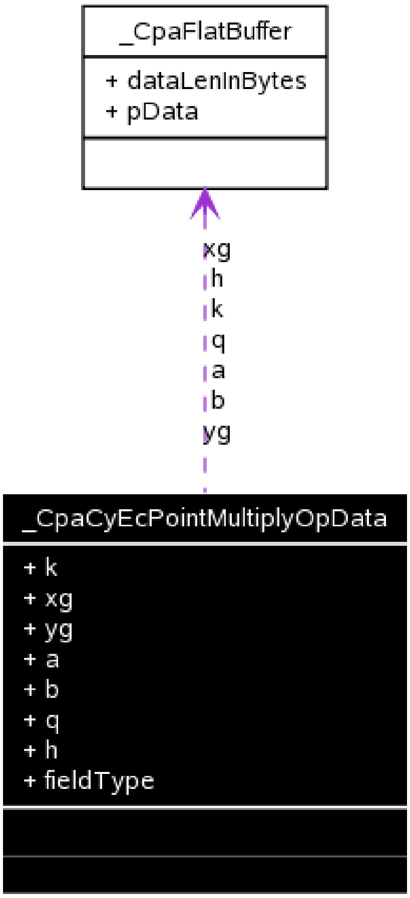
14.8 数据结构文档

14.8.1 _CpaCyEcPointMultiplyOpData Struct Reference

14.8.2 _CpaCyEcPointMultiplyOpData 结构引用

Collaboration diagram for _CpaCyEcPointMultiplyOpData:

_CpaCyEcPointMultiplyOpData 的协作图:



14.8.2.1 Detailed Description
14.8.2.2 详细描述

EC Point Multiplication Operation Data.
EC 点乘运算数据。

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.
此结构包含 cpaCyEcPointMultiply 函数的操作数据。客户端必须为这个结构和这个结构指向的项目分配内存。当结构被传递给函数时，内存的所有权就传递给了函数。当这个结构在回调函数中返回时，内存的所有权返回给客户端。

For optimal performance all data buffers SHOULD be 8-byte aligned.
为了获得最佳性能，所有数据缓冲区都应 8 字节对齐。

All values in this structure are required to be in Most Significant Byte first order, e.g. a.pData[0] = MSB.
该结构中的所有值都要求以最高有效字节优先，例如 a.pData[0] = MSB。

Note:
注意:

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
如果客户端在将此结构中引用的内存提交给 cpaCyEcPointMultiply 函数之后，在回调中返回之前，修改或释放该内存，则未定义的行为

14.6.1 _CpaCyEcPointMultiplyOpData Struct Reference

will result.

14.6.2将产生_CpaCyEcPointMultiplyOpData 结构引用。

See also:

另请参见:

cpaCyEcPointMultiply()

cpaCyEcPointMultiply()

14.8.2.3 Data Fields

14.8.2.4 数据字段

- **CpaFlatBuffer k**
- CpaFlatBuffer k
- **CpaFlatBuffer xg**
- CpaFlatBuffer xg
- **CpaFlatBuffer yg**
- CpaFlatBuffer yg
- **CpaFlatBuffer a**
- CpaFlatBuffer a
- **CpaFlatBuffer b**
- CpaFlatBuffer b
- **CpaFlatBuffer q**
- CpaFlatBuffer q
- **CpaFlatBuffer h**
- CpaFlatBuffer h
- **CpaCyEcFieldType fieldType**
- CpaCyEcFieldType fieldType

14.8.2.5 Field Documentation

14.8.2.6 现场文件

CpaFlatBuffer _CpaCyEcPointMultiplyOpData::k

CpaFlatBuffer _CpaCyEcPointMultiplyOpData::k

scalar multiplier (k > 0 and k < n)

标量乘数(k > 0 且 k < n)

CpaFlatBuffer _CpaCyEcPointMultiplyOpData::xg

CpaFlatBuffer _CpaCyEcPointMultiplyOpData::xg

x coordinate of curve point

曲线点的 x 坐标

CpaFlatBuffer _CpaCyEcPointMultiplyOpData::yg

CpaFlatBuffer _CpaCyEcPointMultiplyOpData::yg

y coordinate of curve point

曲线点的 y 坐标

CpaFlatBuffer _CpaCyEcPointMultiplyOpData::a

CpaFlatBuffer _CpaCyEcPointMultiplyOpData::a

a elliptic curve coefficient

椭圆曲线系数

CpaFlatBuffer _CpaCyEcPointMultiplyOpData::b

CpaFlatBuffer _CpaCyEcPointMultiplyOpData::b

b elliptic curve coefficient

b 椭圆曲线系数

CpaFlatBuffer _CpaCyEcPointMultiplyOpData::q

CpaFlatBuffer _CpaCyEcPointMultiplyOpData::q

prime modulus or irreducible polynomial over GF(2^m)
GF(2^m上的素数模或不可约多项式)

CpaFlatBuffer _CpaCyEcPointMultiplyOpData::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.
操作的余因子。如果不需要余因子，则将余因子设置为1或将平面缓冲区的数据指针设置为空。

CpaCyEcFieldType _CpaCyEcPointMultiplyOpData::fieldType
CpaCyEcFieldType _CpaCyEcPointMultiplyOpData::fieldType

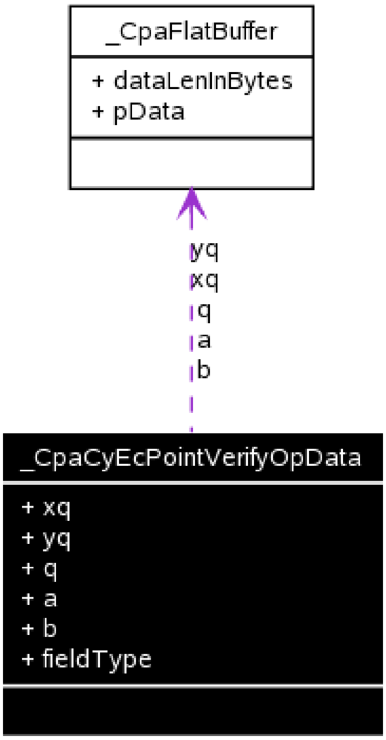
field type for the operation
操作的字段类型

14.6.3 _CpaCyEcPointVerifyOpData Struct Reference

14.6.4 _CpaCyEcPointVerifyOpData结构引用

Collaboration diagram for _CpaCyEcPointVerifyOpData:

_CpaCyEcPointVerifyOpData的协作图:



14.6.4.1 Detailed Description

14.6.4.2 详细描述

EC Point Verification Operation Data.
EC 点验证操作数据。

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.
此结构包含 cpaCyEcPointVerify 函数的操作数据。客户端必须为这个结构和这个结构指向的项目分配内存。当结构被传递给函数时，内存的所有权就传递给了函数。当这个结构在回调函数中返回时，内存的所有权返回给客户端。

For optimal performance all data buffers SHOULD be 8-byte aligned.
为了获得最佳性能，所有数据缓冲区都应 8 字节对齐。

All values in this structure are required to be in Most Significant Byte first order, e.g. a.pData[0] = MSB.
该结构中的所有值都要求以最高有效字节优先，例如 a.pData[0] = MSB。

Note:
注意:

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.

如果客户端在将此结构中引用的内存提交给 CpaCyEcPointVerify 函数之后，在回调中返回之前修改或释放该内存，将导致未定义的行为。

See also:

另请参见:

cpaCyEcPointVerify()

cpaCyEcPointVerify()

14.6.4.3 Data Fields

14.6.4.4 数据字段

- **CpaFlatBuffer xq**
- CpaFlatBuffer xq
- **CpaFlatBuffer yq**
- CpaFlatBuffer yq
- **CpaFlatBuffer q**
- CpaFlatBuffer q
- **CpaFlatBuffer a**
- CpaFlatBuffer a
- **CpaFlatBuffer b**
- CpaFlatBuffer b
- **CpaCyEcFieldType fieldType**
- CpaCyEcFieldType fieldType

14.6.4.5 Field Documentation

14.6.4.6 现场文件

CpaFlatBuffer _CpaCyEcPointVerifyOpData::xg

x coordinate candidate point

x 坐标候选点

CpaFlatBuffer _CpaCyEcPointVerifyOpData::yg

y coordinate candidate point

y 坐标候选点

CpaFlatBuffer _CpaCyEcPointVerifyOpData::q

prime modulus or irreducible polynomial over GF(2^m)

GF(2^m 上的素数模或不可约多项式)

CpaFlatBuffer _CpaCyEcPointVerifyOpData::a

a elliptic curve coefficient

椭圆曲线系数

CpaFlatBuffer _CpaCyEcPointVerifyOpData::b

b elliptic curve coefficient

b 椭圆曲线系数

CpaCyEcFieldType _CpaCyEcPointVerifyOpData::fieldType

field type for the operation

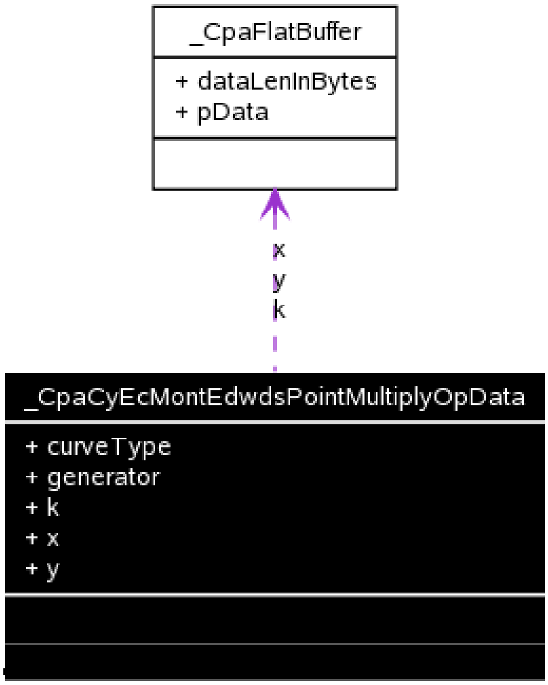
操作的字段类型

14.6.5 _CpaCyEcMontEdwdsPointMultiplyOpData Struct Reference

14.6.6 _ CpaCyEcMontEdwdsPointMultiplyOpData 结构引用

Collaboration diagram for _CpaCyEcMontEdwdsPointMultiplyOpData:

_ CpaCyEcMontEdwdsPointMultiplyOpData 的协作图:



14.6.6.1 Detailed Description
14.6.6.2 详细描述

EC Point Multiplication Operation Data for Edwards or 8 Montgomery curves as specificied in RFC#7748.
RFC#7748 中规定的 Edwards 或 8 Montgomery 曲线的 EC 点乘运算数据。

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
此结构包含 cpaCyEcMontEdwdsPointMultiply 函数的操作数据。客户端必须为这个结构和这个结构指向的项目分配内存。当结构被传递时

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.

到函数中，内存的所有权传递给函数。当这个结构在回调函数中返回时，内存的所有权返回给客户端。

For optimal performance all data buffers SHOULD be 8-byte aligned.
为了获得最佳性能，所有数据缓冲区都应 8 字节对齐。

All values in this structure are required to be in Most Significant Byte first order, e.g. a.pData[0] = MSB.
该结构中的所有值都要求以最高有效字节优先，例如 a.pData[0] = MSB。

Note:
注意:

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.
如果客户端在将此结构中引用的内存提交给 cpaCyEcPointMultiply 函数之后，在回调中返回之前修改或释放该内存，将导致未定义的行为。

All buffers in this structure need to be:
该结构中的所有缓冲器需要:

- 32 bytes in size for 25519 curves
- 25519 条曲线的大小为 32 字节
- 64 bytes in size for 448 curves
- 448 条曲线的大小为 64 字节

See also:
另请参见:

cpaCyEcMontEdwdsPointMultiply()
cpaCyEcMontEdwdsPointMultiply()

14.6.6.3 Data Fields
14.6.6.4 数据字段

- **CpaCyEcMontEdwdsCurveType curveType**
- CpaCyEcMontEdwdsCurveType curveType
- **CpaBoolean generator**
- CpaBoolean generator
- **CpaFlatBuffer k**
- CpaFlatBuffer k
- **CpaFlatBuffer x**
- CpaFlatBuffer x
- **CpaFlatBuffer y**
- CpaFlatBuffer y

14.6.6.5 Field Documentation

14.6.6.6 现场文件

CpaCyEcMontEdwdsCurveType _CpaCyEcMontEdwdsPointMultiplyOpData::curveType

field type for the operation
操作的字段类型

CpaBoolean _CpaCyEcMontEdwdsPointMultiplyOpData::generator

True if the operation is a generator multiplication (kG) False if it is a variable point multiplcation (kP).
如果操作是生成器乘法(kG),则为True如果是可变点乘法(kP),则为False。

CpaFlatBuffer _CpaCyEcMontEdwdsPointMultiplyOpData::k

k or generator for the operation
k或发电机的操作

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
x值。用于标量变量点乘运算。如果生成器为真，则不需要。如果不需要，则必须为NULL。对于25519条曲线，缓冲区的大小必须为32B，对于448条曲线，缓冲区的大小必须为64B

CpaFlatBuffer _CpaCyEcMontEdwdsPointMultiplyOpData::y

y value. Used in variable point multiplication of operations. Not required for curves defined only on scalar 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
y值。用于可变点乘法运算。仅对标量操作定义的曲线不需要。如果生成器为真，则不需要。如果不需要，则必须为NULL。对于25519条曲线，缓冲区的大小必须为32B，对于448条曲线，缓冲区的大小必须为64B

14.6.5 _CpaCyEcStats64 Struct Reference

14.6.6 _CpaCyEcStats64结构引用

14.6.4 _CpaCyEcStats64 Struct Reference

14.6.5 _CpaCyEcStats64 结构引用

14.6.5.1 Detailed Description

14.6.5.2 详细描述

Cryptographic EC Statistics.
加密 EC 统计。

This structure contains statistics on the Cryptographic EC operations. Statistics are set to zero when the component is initialized, and are collected per instance.
此结构包含加密 EC 操作的统计信息。当组件初始化时，统计信息被设置为零，并针对每个实例进行收集。

14.6.5.3 Data Fields

14.6.5.4 数据字段

- **Cpa64U numEcPointMultiplyRequests**
- Cpa64U numEcPointMultiplyRequests
- **Cpa64U numEcPointMultiplyRequestErrors**
- Cpa64U numEcPointMultiplyRequestErrors
- **Cpa64U numEcPointMultiplyCompleted**
- Cpa64U numEcPointMultiplyCompleted
- **Cpa64U numEcPointMultiplyCompletedError**
- Cpa64U numEcPointMultiplyCompletedError
- **Cpa64U numEcPointMultiplyCompletedOutputInvalid**
- Cpa64U numEcPointMultiplyCompletedOutputInvalid
- **Cpa64U numEcPointVerifyRequests**
- Cpa64U numEcPointVerifyRequests
- **Cpa64U numEcPointVerifyRequestErrors**
- Cpa64U numEcPointVerifyRequestErrors
- **Cpa64U numEcPointVerifyCompleted**
- Cpa64U numEcPointVerifyCompleted
- **Cpa64U numEcPointVerifyCompletedErrors**
- Cpa64U numEcPointVerifyCompletedErrors
- **Cpa64U numEcPointVerifyCompletedOutputInvalid**
- Cpa64U numEcPointVerifyCompletedOutputInvalid

14.6.5.5 Field Documentation

14.6.5.6 现场文件

Cpa64U _CpaCyEcStats64::numEcPointMultiplyRequests

Cpa64U _CpaCyEcStats64::numEcPointMultiplyRequests

Total number of EC Point Multiplication operation requests.
EC 点乘操作请求的总数。

Cpa64U _CpaCyEcStats64::numEcPointMultiplyRequestErrors

Cpa64U _CpaCyEcStats64::numEcPointMultiplyRequestErrors

Total number of EC Point Multiplication operation requests that had an error and could not be processed.
有错误且无法处理的 EC 点乘操作请求的总数。

Cpa64U _CpaCyEcStats64::numEcPointMultiplyCompleted

Cpa64U _CpaCyEcStats64::numEcPointMultiplyCompleted

Total number of EC Point Multiplication operation requests that completed successfully.
成功完成的 EC 点乘操作请求的总数。

Cpa64U CpaCyEcStats64::numEcPointMultiplyCompletedError

Total number of EC Point Multiplication operation requests that could not be completed successfully due to errors.
由于错误而无法成功完成的 EC 点乘操作请求的总数。

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.
由于无效输出而无法成功完成的 EC 点乘操作请求的总数。请注意，这并不表示有错误。

Cpa64U CpaCyEcStats64::numEcPointVerifyRequests

Total number of EC Point Verification operation requests.
EC 点验证操作请求的总数。

Cpa64U CpaCyEcStats64::numEcPointVerifyRequestErrors

Total number of EC Point Verification operation requests that had an error and could not be processed.
有错误且无法处理的 EC 点验证操作请求的总数。

Cpa64U CpaCyEcStats64::numEcPointVerifyCompleted

Total number of EC Point Verification operation requests that completed successfully.
成功完成的 EC 点验证操作请求的总数。

Cpa64U CpaCyEcStats64::numEcPointVerifyCompletedErrors

Total number of EC Point Verification operation requests that could not be completed successfully due to errors.
由于错误而无法成功完成的 EC 点验证操作请求的总数。

Cpa64U CpaCyEcStats64::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.
具有无效输出的 EC 点验证操作请求的总数。请注意，这并不表示有错误。

14.9 Typedef Documentation

14.10 Typedef 文档

typedef enum _CpaCyEcFieldType **CpaCyEcFieldType**

typedef enum _CpaCyEcFieldType CpaCyEcFieldType

Field types for Elliptic Curve
椭圆曲线的字段类型

As defined by FIPS-186-3, for each cryptovariable length, there are two kinds of fields.
根据 FIPS-186-3 的定义，对于每个加密变量长度，有两种字段。

- 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.
- 素数域是包含素数 p 个元素的域 GF(p)。这个字段的元素是以 p 为模的整数，字段算术是根据以 p 为模的整数的算术实现的。
- A binary field is the field GF(2^m) which contains 2^m elements for some m (called the degree of
- 二进制域是 GF(2^m 域，它包含某些 m 的 2^m 元素(称为度
  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.
  场)。该字段的元素是长度为 m 的比特串，并且字段算术是根据比特上的运算来实现的。

typedef enum _CpaCyEcMontEdwdsCurveType **CpaCyEcMontEdwdsCurveType**

typedef enum _CpaCyEcMontEdwdsCurveType CpaCyEcMontEdwdsCurveType

Curve types for Elliptic Curves defined in RFC#7748
RFC#7748 中定义的椭圆曲线的曲线类型

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
根据 RFC 7748 的定义，该组中有四条椭圆曲线。蒙哥马利曲线表示为曲线 25519 和曲线 448，双有理等价扭曲爱德华兹曲线表示为爱德华 25519 和爱德华 448

typedef struct _CpaCyEcPointMultiplyOpData **CpaCyEcPointMultiplyOpData**

typedef struct _CpaCyEcPointMultiplyOpData CpaCyEcPointMultiplyOpData

EC Point Multiplication Operation Data.
EC 点乘运算数据。

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.
此结构包含 cpaCyEcPointMultiply 函数的操作数据。客户端必须为这个结构和这个结构指向的项目分配内存。当结构被传递给函数时，内存的所有权就传递给了函数。当这个结构在回调函数中返回时，内存的所有权返回给客户端。

For optimal performance all data buffers SHOULD be 8-byte aligned.
为了获得最佳性能，所有数据缓冲器都应 8 字节对齐。

All values in this structure are required to be in Most Significant Byte first order, e.g. a.pData[0] = MSB.
该结构中的所有值都要求以最高有效字节优先，例如 a.pData[0] = MSB。

Note:
注意：

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.
如果客户端在将此结构中引用的内存提交给 cpaCyEcPointMultiply 函数之后，在回调中返回之前修改或释放该内存，将导致未定义的行为。

See also:
另请参见：

cpaCyEcPointMultiply()
cpaCyEcPointMultiply()

typedef struct _**CpaCyEcPointVerifyOpData CpaCyEcPointVerifyOpData**

EC Point Verification Operation Data.
EC 点验证操作数据。

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.
此结构包含 cpaCyEcPointVerify 函数的操作数据。客户端必须为这个结构和这个结构指向的项目分配内存。当结构被传递给函数时，内存的所有权就传递给了函数。当这个结构在回调函数中返回时，内存的所有权返回给客户端。

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

为了获得最佳性能，所有数据缓冲器都应 8 字节对齐。

All values in this structure are required to be in Most Significant Byte first order, e.g. a.pData[0] = MSB.
该结构中的所有值都要求以最高有效字节优先，例如 a.pData[0] = MSB。

Note:
注意:

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.
如果客户端在将此结构中引用的内存提交给 CpaCyEcPointVerify 函数之后，在回调中返回之前修改或释放该内存，将导致未定义的行为。

See also:
另请参见:

cpaCyEcPointVerify()
cpaCyEcPointVerify()

typedef struct **_CpaCyEcMontEdwdsPointMultiplyOpData CpaCyEcMontEdwdsPointMultiplyOpData**
typedef 结构_CpaCyEcMontEdwdsPointMultiplyOpDatCpaCyEcMontEdwdsPointMultiplyOpDat

EC Point Multiplication Operation Data for Edwards or 8 Montgomery curves as specificied in RFC#7748.
RFC#7748 中规定的 Edwards 或 8 Montgomery 曲线的 EC 点乘运算数据。

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.
此结构包含 cpaCyEcMontEdwdsPointMultiply 函数的操作数据。客户端必须为这个结构和这个结构指向的项目分配内存。当结构被传递给函数时，内存的所有权就传递给了函数。当这个结构在回调函数中返回时，内存的所有权返回给客户端。

For optimal performance all data buffers SHOULD be 8-byte aligned.
为了获得最佳性能，所有数据缓冲区都应 8 字节对齐。

All values in this structure are required to be in Most Significant Byte first order, e.g. a.pData[0] = MSB.
该结构中的所有值都要求以最高有效字节优先，例如 a.pData[0] = MSB。

Note:
注意:

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.
如果客户端在将此结构中引用的内存提交给 cpaCyEcPointMultiply 函数之后，在回调中返回之前修改或释放该内存，将导致未定义的行为。

All buffers in this structure need to be:

该结构中的所有缓冲器需要:

- 32 bytes in size for 25519 curves
- 25519 条曲线的大小为 32 字节
- 64 bytes in size for 448 curves
- 448 条曲线的大小为 64 字节

See also:
另请参见:

cpaCyEcMontEdwdsPointMultiply()
cpaCyEcMontEdwdsPointMultiply()

typedef struct _**CpaCyEcStats64 CpaCyEcStats64**
typedef 结构_CpaCyEcStats6CpaCyEcStats6

Cryptographic EC Statistics.
加密 EC 统计。

This structure contains statistics on the Cryptographic EC operations. Statistics are set to zero when the component is initialized, and are collected per instance.
此结构包含加密 EC 操作的统计信息。当组件初始化时，统计信息被设置为零，并针对每个实例进行收集。

typedef void(* **CpaCyEcPointMultiplyCbFunc**)(void *pCallbackTag, **CpaStatus** status, void *pOpData
typedef void(* **CpaCyEcPointMultiplyCbFunc**)(void *pCallbackTag, **CpaStatus** status, void *pOp

Definition of callback function invoked for cpaCyEcPointMultiply requests.
为 cpaCyEcPointMultiply 请求调用的回调函数的定义。

Context:
背景:

This callback function can be executed in a context that DOES NOT permit sleeping to occur.
这个回调函数可以在不允许休眠发生的上下文中执行。

Assumptions:
假设:

None
没有人

Side-Effects:
副作用:

None
没有人

Reentrant:

可重入:

No

不

Thread-safe:

线程安全:

Yes

是

Parameters:

参数:

`[in]` *pCallbackTag* User-supplied value to help identify request.

[in] pCallbackTag 使用者提供的值，可协助识别要求。

`[in]` *status* Status of the operation. Valid values are CPA_STATUS_SUCCESS, CPA_STATUS_FAIL and CPA_STATUS_UNSUPPORTED.

[in] status 操作的状态。有效值为 CPA_STATUS_SUCCESS、CPA_STATUS_FAIL 和 CPA_STATUS_UNSUPPORTED。

`[in]` *pOpData* Opaque pointer to Operation data supplied in request.

[in]指向请求中提供的操作数据的 pOpData Opaque 指针。

`[in]` *multiplyStatus* Status of the point multiplication.

[in]点乘法的多状态状态。

`[in]` *pXk* x coordinate of resultant EC point.

[in]生成的 EC 点的 pXk x 坐标。

`[in]` *pYk* y coordinate of resultant EC point.

[in]生成的 EC 点的 pYk y 坐标。

Return values:

返回值:

None

没有人

Precondition:

前提条件:

Component has been initialized.

组件已初始化。

Postcondition:

后置条件:

None

没有人

Note:

注意:

None

没有人

See also:

另请参见:

cpaCyEcPointMultiply()

cpaCyEcPointMultiply()

typedef void(* **CpaCyEcPointVerifyCbFunc**)(void *pCallbackTag, **CpaStatus** status, void *pOpData,

typedef void(* **CpaCyEcPointVerifyCbFunc**)(void *pCallbackTag, **CpaStatus** status, void *pOpDa

Definition of callback function invoked for cpaCyEcPointVerify requests.

为 cpaCyEcPointVerify 请求调用的回调函数的定义。

Context:

背景:

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

这个回调函数可以在不允许休眠发生的上下文中执行。

Assumptions:

假设:

None

没有人

Side-Effects:

副作用:

None

没有人

Reentrant:

可重入:

No

不

Thread-safe:

线程安全:

Yes

是

Parameters:

参数:

[in] *pCallbackTag* User-supplied value to help identify request.

[in] pCallbackTag 使用者提供的值，可协助识别要求。

[in] *status* Status of the operation. Valid values are CPA_STATUS_SUCCESS, CPA_STATUS_FAIL and CPA_STATUS_UNSUPPORTED.

[in] status 操作的状态。有效值为 CPA_STATUS_SUCCESS、CPA_STATUS_FAIL 和 CPA_STATUS_UNSUPPORTED。

[in] *pOpData* Operation data pointer supplied in request.

[in]请求中提供了 pOpData 操作数据指针。

[in] *verifyStatus*

[in]验证状态

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.

如果点不在曲线上或在无穷远处，则设置为CPA_FALSE。如果点在曲线上，则设置为CPA_TRUE。

Returns:

退货:

None
没有人

Precondition:
前提条件:

Component has been initialized.
组件已初始化。

Postcondition:
后置条件:

None
没有人

Note:

注意:

None
没有人

See also:

另请参见:

cpaCyEcPointVerify()
cpaCyEcPointVerify()

14.8 Enumeration Type Documentation

14.9 枚举类型文档

enum **CpaCyEcFieldType**
CpaCyEcFieldT

Field types for Elliptic Curve
椭圆曲线的字段类型

As defined by FIPS-186-3, for each cryptovariable length, there are two kinds of fields.
根据FIPS-186-3的定义，对于每个加密变量长度，有两种字段。

- 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.
- 素数域是包含素数p个元素的域GF(p)。这个字段的元素是以p为模的整数，字段算术是根据以p为

模的整数的算术实现的。

- A binary field is the field GF(2^m) which contains 2^m elements for some m (called the degree of
- 二进制域是 GF(2ˆm 域，它包含某些 m 的 2ˆm 元素(称为度

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.

场)。该字段的元素是长度为 m 的比特串，并且字段算术是根据比特上的运算来实现的。

Enumerator:
枚举器:

CPA_CY_EC_FIELD_TYPE_PRIME A prime field, GF(p)
CPA_CY_EC_FIELD_TYPE_PRIME 一个素数域 GF(p)
CPA_CY_EC_FIELD_TYPE_BINARY A binary field, GF(2^m)
CPA_CY_EC_FIELD_TYPE_BINARY 二进制字段，GF(2ˆm)

enum CpaCyEcMontEdwdsCurveType

Curve types for Elliptic Curves defined in RFC#7748
RFC#7748 中定义的椭圆曲线的曲线类型

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
根据 RFC 7748 的定义，该组中有四条椭圆曲线。蒙哥马利曲线表示为曲线 25519 和曲线 448，双有理等价扭曲爱德华兹曲线表示为爱德华 25519 和爱德华 448

Enumerator:
枚举器:

CPA_CY_EC_MONTEDWDS_CURVE25519_TYPE	Montgomery 25519 curve
CPA_CY_EC_MONTEDWDS_ED25519_TYPE	Twisted Edwards 25519 curve
CPA_CY_EC_MONTEDWDS_CURVE448_TYPE	Montgomery 448 curve
CPA_CY_EC_MONTEDWDS_ED448_TYPE	Twisted Edwards 448 curve

CPA _ CY _ EC _ monted WDS _ curve 25519 _ TYPE 蒙哥马利 25519 曲线 CPA _ CY _ EC _ monted WDS _ ed 25519 _ TYPE 扭曲的 Edwards 25519 曲线
CPA _ CY _ EC _ monted WDS _ curve 448 _ TYPE 蒙哥马利 448 曲线
CPA_CY_EC_MONTEDWDS_ED448_TYPE 扭曲的爱德华兹 448 曲线

14.10 Function Documentation

14.11 功能文档

cpaCyEcPointMultiply (const instanceHandle

```
cpaCyEcPointMultiply  (  const                                      instanceHandle,
                              const pCb,
                              void *pCallbackTag,
                              const * pOpData,
                              *pMultiplyStatus,
                              *pXk,
                              *pYk
                            )
```

Perform EC Point Multiplication.
执行 EC 点乘。

This function performs Elliptic Curve Point Multiplication as per ANSI X9.63 Annex D.3.2.
该函数根据 ANSI X9.63 附录 D.3.2 执行椭圆曲线点乘。

Context:
背景:

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.
当作为异步函数调用时，它不能休眠。它可以在不允许休眠的上下文中执行。当作为同步函数调用时，它可能会休眠。它不能在不允许休眠的上下文中执行。

Assumptions:
假设:

None
没有人

Side-Effects:
副作用:

None
没有人

Blocking:
阻止:

Yes when configured to operate in synchronous mode.
当配置为在同步模式下运行时，是。

Reentrant:
可重入:

No
不

Thread-safe:

线程安全:

Yes

是

Parameters:

参数:

[in] *instanceHandle* Instance handle.

[in] instanceHandle 执行个体控制代码。

[in] *pCb* Callback function pointer. If this is set to a NULL value the function will operate synchronously.

[in] pCb 回调函数指针。如果设置为空值，函数将同步运行。

[in] *pCallbackTag* User-supplied value to help identify request.

[in] pCallbackTag 使用者提供的值，可协助识别要求。

[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.

[in] pOpData 结构，包含执行作业所需的所有资料。客户端代码为此结构分配内存。该组件取得内存的所有权，直到它在回调中被返回。

[out] *pMultiplyStatus* In synchronous mode, the multiply output is valid (CPA_TRUE) or the output is invalid (CPA_FALSE).

[out] pMultiplyStatus 在同步模式下，乘法输出有效(CPA_TRUE)或输出无效(CPA_FALSE)。

[out] *pXk* Pointer to xk flat buffer.

[out]指向 Xk 平面缓冲区的 pXk 指针。

[out] *pYk* Pointer to yk flat buffer.

[out]指向 Yk 平面缓冲区的 pYk 指针。

Return values:

返回值:

CPA_STATUS_SUCCESS Function executed successfully.

CPA_STATUS_SUCCESS 函数执行成功。

CPA_STATUS_FAIL Function failed.

CPA_STATUS_FAIL 函数失败。

CPA_STATUS_RETRY Resubmit the request.

CPA_STATUS_INVALID_PARAM Invalid parameter in.

CPA_STATUS_RESOURCE Error related to system resources.

CPA_STATUS_RETRY 重新提交请求。CPA_STATUS_INVALID_PARAM 中的参数无效。与系统资源相关的 CPA_STATUS_RESOURCE 错误。

CPA_STATUS_RESTARTING API implementation is restarting. Resubmit the request.

CPA_STATUS_RESTARTING API 实现正在重新启动。重新提交请求。

CPA_STATUS_UNSUPPORTED Function is not supported.

不支持 CPA_STATUS_UNSUPPORTED 函数。

Precondition:

前提条件:

The component has been initialized via cpaCyStartInstance function.

该组件已通过 cpaCyStartInstance 函数初始化。

Postcondition:

后置条件:

None

没有人

Note:

注意:

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.
当pCb为非空时，会生成一个CpaCyEcPointMultiplyCbFunc类型的异步回调来响应此函数调用。为了获得最佳性能，数据指针应该8字节对齐。

See also:

另请参见:

CpaCyEcPointMultiplyOpData, **CpaCyEcPointMultiplyCbFunc**
CpaCyEcPointMultiplyOpData,CpaCyEcPointMultiplyCbFunc

cpaCyEcPointVerify (const instanceHandle

```
cpaCyEcPointVerify  (  const                                       instanceHandle,
                           const pCb,
                           void *pCallbackTag,
                           const * pOpData,
                           *pVerifyStatus
                          )
```

Verify that a point is on an elliptic curve.
验证一个点在椭圆曲线上。

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**.)
该函数根据ANSI X9.62附录A.4.2的步骤a、b和c执行椭圆曲线点验证。(要执行最后一步d，用户可以调用**cpaCyEcPointMultiply**

This function checks if the specified point satisfies the Weierstrass equation for an Elliptic Curve.
此函数检查指定点是否满足椭圆曲线的Weierstrass方程。

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)
对于GF(p): y^2 = (x^3 + ax + b) mod p对于GF(2^m): y^2 + xy = x^3 + ax^2 + b mod p其中p是GF(2^m上的不可约多项式)

Use this function to verify a point is in the correct range and is NOT the point at infinity.
使用此功能验证一个点是否在正确的范围内，并且不是无穷远处的点。

Context:
背景:

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.
当作为异步函数调用时，它不能休眠。它可以在不允许休眠的上下文中执行。当作为同步函数调用时，它可能会休眠。它不能在不允许休眠的上下文中执行。

Assumptions:
假设:

None
没有人

Side-Effects:
副作用：
None
没有人

Blocking:
阻止：
Yes when configured to operate in synchronous mode.
当配置为在同步模式下运行时，是。

Reentrant:
可重入：
No
不

Thread-safe:
线程安全：
Yes
是

Parameters:
参数：

`[in]` *instanceHandle* Instance handle.

[in] instanceHandle 执行个体控制代码。

`[in]` *pCb* Callback function pointer. If this is set to a NULL value the function will operate synchronously.

[in] pCb 回调函数指针。如果设置为空值，函数将同步运行。

`[in]` *pCallbackTag* User-supplied value to help identify request.

[in] pCallbackTag 使用者提供的值，可协助识别要求。

`[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.

[in] pOpData 结构，包含执行作业所需的所有资料。客户端代码为此结构分配内存。该组件取得内存的所有权，直到它在回调中被返回。

`[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.

[out] pVerifyStatus 在同步模式下，如果点不在曲线上或不在无穷远处，则设置为 CPA_FALSE。如果点在曲线上，则设置为 CPA_TRUE。

Return values:

返回值:

CPA_STATUS_SUCCESS Function executed successfully.

CPA_STATUS_SUCCESS 函数执行成功。

CPA_STATUS_FAIL Function failed.

CPA_STATUS_FAIL 函数失败。

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.

CPA_STATUS_RETRY 重新提交请求。传递的 CPA_STATUS_INVALID_PARAM 参数无效。与系统资源相关的 CPA_STATUS_RESOURCE 错误。CPA_STATUS_RESTARTING API 实现正在重新启动。重新提交请求。不支持 CPA_STATUS_UNSUPPORTED 函数。

Precondition:

前提条件:

The component has been initialized via cpaCyStartInstance function.

该组件已通过 cpaCyStartInstance 函数初始化。

Postcondition:

后置条件:

None

没有人

Note:

注意:

W
h
e
n

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.
当pCb为非空时，会生成一个CpaCyEcPointVerifyCbFunc类型的异步回调来响应此函数调用。为了获得最佳性能，数据指针应该8字节对齐。

See also:

另请参见:

CpaCyEcPointVerifyOpData, **CpaCyEcPointVerifyCbFunc**
CpaCyEcPointVerifyOpData, CpaCyEcPointVerifyCbFunc

```
                                                                              instanceHandle,  ,
cpaCyEcMontEdwdsPointMultiply         (  const                                pCb,  pCallbackTag,
                                      const
                                      void *
                                         const
                                           pOpData,
                                         *
                                         *pMultiplyStatus,
                                         *pXk,
                                         *pYk
                                      )
```

Perform EC Point Multiplication on an Edwards or Montgomery curve as defined in RFC#7748.

This function performs Elliptic Curve Point Multiplication as per RFC#7748

按照RFC#7748中的定义，在Edwards或Montgomery曲线上执行EC点乘法。该函数根据

RFC#7748执行椭圆曲线点乘

Context:
背景:

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.
当作为异步函数调用时，它不能休眠。它可以在不允许休眠的上下文中执行。当作为同步函数调用时，它可能会休眠。它不能在不允许休眠的上下文中执行。

Assumptions:
假设:

None
没有人

Side-Effects:

副作用:

None

没有人

Blocking:

阻止:

Yes when configured to operate in synchronous mode.

当配置为在同步模式下运行时，是。

Reentrant:

可重入:

No

不

Thread-safe:

线程安全:

Yes

是

Parameters:

参数:

`[in]` *instanceHandle* Instance handle.

[in] instanceHandle 执行个体控制代码。

`[in]` *pCb* Callback function pointer. If this is set to a NULL value the function will operate synchronously.

[in] pCb 回调函数指针。如果设置为空值，函数将同步运行。

`[in]` *pCallbackTag* User-supplied value to help identify request.

[in] pCallbackTag 使用者提供的值，可协助识别要求。

`[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.

[in] pOpData 结构，包含执行作业所需的所有资料。客户端代码为此结构分配内存。该组件取得内存的所有权，直到它在回调中被返回。

`[out]` *pMultiplyStatus* In synchronous mode, the multiply output is valid (CPA_TRUE) or the output is invalid (CPA_FALSE).

[out] pMultiplyStatus 在同步模式下，乘法输出有效(CPA_TRUE)或输出无效(CPA_FALSE)。

`[out]` *pXk* Pointer to xk flat buffer.

[out]指向 Xk 平面缓冲区的 pXk 指针。

`[out]` *pYk* Pointer to yk flat buffer.

[out]指向 Yk 平面缓冲区的 pYk 指针。

Return values:

返回值:

CPA_STATUS_SUCCESS Function executed successfully.

CPA_STATUS_SUCCESS 函数执行成功。

CPA_STATUS_FAIL Function failed.

CPA_STATUS_FAIL 函数失败。

CPA_STATUS_RETRY Resubmit the request.

CPA_STATUS_INVALID_PARAM Invalid parameter in.

CPA_STATUS_RESOURCE Error related to system resources.

CPA_STATUS_RETRY 重新提交请求。CPA_STATUS_INVALID_PARAM 中的参数无效。与系统资源相关的 CPA_STATUS_RESOURCE 错误。

CPA_STATUS_RESTARTING API implementation is restarting. Resubmit the request.

CPA_STATUS_RESTARTING API 实现正在重新启动。重新提交请求。

CPA_STATUS_UNSUPPORTED Function is not supported.

不支持 CPA_STATUS_UNSUPPORTED 函数。

Precondition:

前提条件:

The component has been initialized via cpaCyStartInstance function.

该组件已通过 cpaCyStartInstance 函数初始化。

Postcondition:

后置条件:

None

没有人

Note:

注意:

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.

当 pCb 为非空时，会生成一个 CpaCyEcPointMultiplyCbFunc 类型的异步回调来响应此函数调用。为了获得最佳性能，数据指针应该 8 字节对齐。

See also:

另请参见:

CpaCyEcMontEdwdsPointMultiplyOpData, CpaCyEcMontEdwdsPointMultiplyCbFunc

CpaCyEcMontEdwdsPointMultiplyOpData，CpaCyEcMontEdwdsPointMultiplyCbFunc

```
CpaStatus cpaCyEcQueryStats64 ( const CpaInstanceHandle instanceHandle,
                                CpaCyEcStats64 *        pEcStats
```

```
CpaStatus cpaCyEcQueryStats64 ( const CpaInstanceHandle instanceHandle,
                              ` CpaCyEcStats64 *        pEcStats
                              `
```

Query statistics for a specific EC instance.

查询特定EC实例的统计信息。

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.
该函数将查询EC实现的特定实例的统计信息。用户必须分配CpaCyEcStats64结构，并将对该结构的引用传递到此函数调用中。此函数将统计结果写入传入的CpaCyEcStats64结构中。

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.
注意:此函数返回的统计数据不会中断当前的数据处理，因此可能会与统计数据检索过程中正在进行的操作稍微不同步。

Context:
背景:

This is a synchronous function and it can sleep. It MUST NOT be executed in a context that DOES NOT permit sleeping.
这是一个同步功能，它可以休眠。它不能在不允许休眠的上下文中执行。

Assumptions:
假设:

None
没有人

Side-Effects:
副作用:

None
没有人

Blocking:
阻止:

This function is synchronous and blocking.
这个函数是同步的和阻塞的。

Reentrant:
可重入:

No
不

Thread-safe:
线程安全:

Yes
是

Parameters:
参数:

[in] *instanceHandle* Instance handle.

[in] instanceHandle 执行个体控制代码。

[out] *pEcStats* Pointer to memory into which the statistics will be written.

[out]指向将写入统计信息的内存的 pEcStats 指针。

Return values:

返回值:

CPA_STATUS_SUCCESS Function executed successfully.

CPA_STATUS_SUCCESS 函数执行成功。

CPA_STATUS_FAIL Function failed.

CPA_STATUS_INVALID_PARAM Invalid parameter passed in.

CPA_STATUS_RESOURCE Error related to system resources.

CPA_STATUS_FAIL 函数失败。传递的 CPA_STATUS_INVALID_PARAM 参数无效。与系统资源相关的 CPA_STATUS_RESOURCE 错误。

CPA_STATUS_RESTARTING API implementation is restarting. Resubmit the request.

CPA_STATUS_RESTARTING API 实现正在重新启动。重新提交请求。

CPA_STATUS_UNSUPPORTED Function is not supported.

不支持 CPA_STATUS_UNSUPPORTED 函数。

Precondition:

前提条件:

Component has been initialized.

组件已初始化。

Postcondition:

后置条件:

None

没有人

Note:

注意:

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

该函数以同步方式运行，不会生成异步回调。

See also:

另请参见:

CpaCyEcStats64

CpaCyEcStats64

```
cpaCyKptEcPointMultiply (  const                                    instanceHandle,
                          const pCb,
                          void *pCallbackTag,
                          const *  pOpData,
                          *pMultiplyStatus,
                          *pXk,
                          *pYk,
                          *pKptUnwrapContext
                         )
```

```
cpaCyKptEcPointMultiply (  const                                    instanceHandle,
                          const pCb,
                          void *pCallbackTag,
                          const *  pOpData,
                          *pMultiplyStatus,
                          *pXk,
                          *pYk,
                          *pKptUnwrapContext
                         )
```

Perform KPT mode EC Point Multiplication.
执行 KPT 模式 EC 点乘。

This function is variant of cpaCyEcPointMultiply, which will perform Elliptic Curve Point Multiplication as per ANSI X9.63 Annex D.3.2.
此函数是 cpaCyEcPointMultiply 的变体，它将根据 ANSI X9.63 附录 D.3.2 执行椭圆曲线点乘法。

Context:
背景:

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.
当作为异步函数调用时，它不能休眠。它可以在不允许休眠的上下文中执行。当作为同步函数调用时，它可能会休眠。它不能在不允许休眠的上下文中执行。

Assumptions:
假设:

None
没有人

Side-Effects:
副作用:

None
没有人

Blocking:
阻止:

Yes when configured to operate in synchronous mode.
当配置为在同步模式下运行时，是。

Reentrant:
可重入:

No
不

Thread-safe:
线程安全:
Yes
是

Parameters:
参数:

[in] *instanceHandle* Instance handle.
[in] instanceHandle 执行个体控制代码。

[in] *pCb* Callback function pointer. If this is set to a NULL value the function
[in] pCb 回调函数指针。如果设置为空值，函数
will operate synchronously.
将同步运行。

[in] *pCallbackTag* User-supplied value to help identify request.
[in] pCallbackTag 使用者提供的值，可协助识别要求。

[in] *pOpData* Structure containing all the data needed to perform the operation.
[in] pOpData 结构，包含执行作业所需的所有资料。
The client code allocates the memory for this structure. This component takes ownership of the memory until it is returned in the callback.
客户端代码为此结构分配内存。该组件取得内存的所有权，直到它在回调中被返回。

[out] *pMultiplyStatus* In synchronous mode, the multiply output is valid (CPA_TRUE) or the
[out] pMultiplyStatus 在同步模式下，乘法输出有效(CPA_TRUE)或
output is invalid (CPA_FALSE).
输出无效(CPA_FALSE)。

[out] *pXk* Pointer to xk flat buffer.
[out]指向 Xk 平面缓冲区的 pXk 指针。

[out] *pYk* Pointer to yk flat buffer.
[out]指向 Yk 平面缓冲区的 pYk 指针。

[in] *pKptUnwrapContext* Pointer of structure into which the content of KptUnwrapContext is
[in]结构的 pKptUnwrapContext 指标，KptUnwrapContext 的内容会放入其中
kept,The client MUST allocate this memory and copy structure KptUnwrapContext into this flat buffer.
保持，客户端必须分配这个内存并将结构 KptUnwrapContext 复制到这个平面缓冲区中。

Return values:
返回值:

CPA_STATUS_SUCCESS Function executed successfully.
CPA_STATUS_SUCCESS 函数执行成功。

CPA_STATUS_FAIL Function failed.
CPA_STATUS_FAIL 函数失败。

CPA_STATUS_RETRY Resubmit the request.
CPA_STATUS_RETRY 重新提交请求。

CPA_STATUS_INVALID_PARAM Invalid parameter in.

CPA_STATUS_INVALID_PARAM 中的参数无效。

CPA_STATUS_RESOURCE Error related to system resources.

与系统资源相关的 CPA_STATUS_RESOURCE 错误。

CPA_STATUS_RESTARTING API implementation is restarting. Resubmit the request.

CPA_STATUS_RESTARTING API 实现正在重新启动。重新提交请求。

Precondition:
前提条件:

The component has been initialized via cpaCyStartInstance function.

该组件已通过 cpaCyStartInstance 函数初始化。

Postcondition:
后置条件:

None

没有人

Note:

注意:

By virtue of invoking the cpaCyKptEcPointMultiply, the implementation understands that CpaCyEcPointMultiplyOpData contains an encrypted private key that requires unwrapping. KptUnwrapContext contains an 'KptHandle' field that points to the unwrapping key in the WKT. When pCb is non-NULL an asynchronous callback of type CpaCyEcPointMultiplyCbFunc is generated in response to this function call. In KPT release,private key field in cpaCyKptEcPointMultiply is a concatenation of cipher text and hash tag. For optimal performance, data pointers SHOULD be 8-byte aligned.

通过调用cpaCyKptEcPointMultiply，该实现了解到CpaCyEcPointMultiplyOpData包含需要解包的加密私钥。KptUnwrapContext包含一个指向WKT中的展开密钥的“KptHandle”字段。当pCb为非空时，会生成一个CpaCyEcPointMultiplyCbFunc类型的异步回调来响应此函数调用。在KPT版本中，cpaCyKptEcPointMultiply中的私钥字段是密文和散列标签的串联。为了获得最佳性能，数据指针应该8字节对齐。

See also:

另请参见:

CpaCyEcPointMultiplyOpData, **CpaCyEcPointMultiplyCbFunc**

CpaCyEcPointMultiplyOpData, CpaCyEcPointMultiplyCbFunc

16 Elliptic Curve Diffie-Hellman (ECDH) API

17 椭圆曲线 Diffie-Hellman (ECDH) API

[Cryptographic API]

[Cryptographic API]

Collaboration diagram for Elliptic Curve Diffie-Hellman (ECDH) API:
椭圆曲线 Diffie-Hellman (ECDH) API 协作图:

Cryptographic API
Elliptic Curve Diffie-Hellman (ECDH) API
Cryptographic API
Elliptic Curve Diffie-Hellman (ECDH) API

17.1 Detailed Description

17.2 详细描述

File: cpa_cy_ecdh.h

文件:cpa_cy_ecdh.h

These functions specify the API for Public Key Encryption (Cryptography) Elliptic Curve Diffie-Hellman (ECDH) operations.
这些函数指定了公钥加密(加密)椭圆曲线 Diffie-Hellman (ECDH)操作的 API。

Note:
注意:

Large numbers are represented on the QuickAssist API as described in the Large Number API (**Cryptographic Large Number API**).
大数在 QuickAssist API 上表示，如大数 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.
此外，对于椭圆曲线运算，传递给 API 的大数的位长度不得超过 576 位。

17.3 Data Structures

17.4 数据结构

- struct **_CpaCyEcdhPointMultiplyOpData**

- 结构体_CpaCyEcdhPointMultiplyOpData

- struct **_CpaCyEcdhStats64**
- 结构体_CpaCyEcdhStats64

17.5 Typedefs
17.6 类型定义

- typedef **_CpaCyEcdhPointMultiplyOpData CpaCyEcdhPointMultiplyOpData**

- 数据类型说明_CpaCyEcdhPointMultiplyOpData CpaCyEcdhPointMultiplyOpData
- typedef **_CpaCyEcdhStats64 CpaCyEcdhStats64**
- 数据类型说明_CpaCyEcdhStats64 CpaCyEcdhStats64
- typedef void(* **CpaCyEcdhPointMultiplyCbFunc**)(void *pCallbackTag, **CpaStatus** status, void
- typedef void(*CpaCyEcdhPointMultiplyCbFunc CpaStatus
  *pOpData, **CpaBoolean** multiplyStatus, **CpaFlatBuffer** *pXk, **CpaFlatBuffer** *pYk)
  *pOpData，CpaBoolean CpaFlatBuffer CpaFlatBuffer

17.7 Functions
17.8 功能

- **CpaStatus cpaCyEcdhPointMultiply** (const **CpaInstanceHandle** instanceHandle, const **CpaCyEcdhPointMultiplyCbFunc** pCb, void *pCallbackTag, const **CpaCyEcdhPointMultiplyOpData** *pOpData, **CpaBoolean** *pMultiplyStatus, **CpaFlatBuffer** *pXk, **CpaFlatBuffer** *pYk)

- CpaStatus cpaCyEcdhPointMultiply （常量CpaInstanceHandle CpaCyEcdhPointMultiplyCbFunc CpaCyEcdhPointMultiplyOpData CpaBoolean CpaFlatBuffer CpaFlatBuffer
- **CpaStatus cpaCyEcdhQueryStats64** (const **CpaInstanceHandle** instanceHandle,
- CpaStatus cpaCyEcdhQueryStats64 （常量CpaInstanceHandle
  CpaCyEcdhStats64 *pEcdhStats)
  CpaCyEcdhStats64 *pEcdhStats)

17.9 Data Structure Documentation
17.10 数据结构文档

15.5 Data Structure Documentation

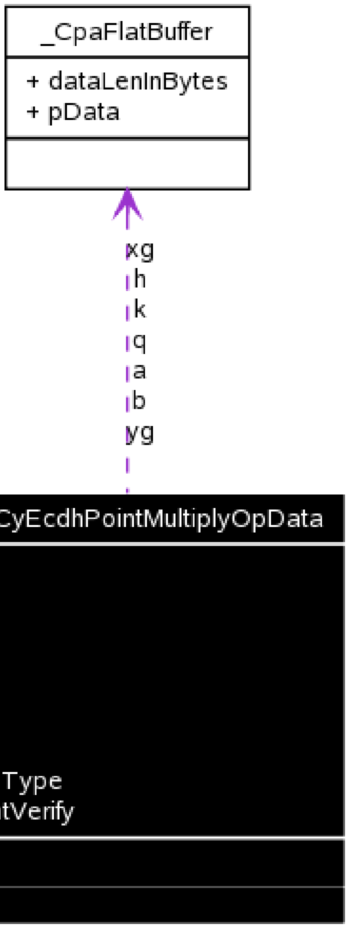
15.6 数据结构文档

15.6.1 _CpaCyEcdhPointMultiplyOpData Struct Reference

15.6.2 _CpaCyEcdhPointMultiplyOpData 结构引用

Collaboration diagram for _CpaCyEcdhPointMultiplyOpData:

_ CpaCyEcdhPointMultiplyOpData 的协作图:



15.6.2.1 Detailed Description
15.6.2.2 详细描述

ECDH Point Multiplication Operation Data.
ECDH 点乘运算数据。

This structure contains the operation data for the cpaCyEcdhPointMultiply 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.
此结构包含 cpaCyEcdhPointMultiply 函数的操作数据。客户端必须为这个结构和这个结构指向的项目分配内

存。当结构被传递给函数时，内存的所有权就传递给了函数。当这个结构在回调函数中返回时，内存的所有权返回给客户端。

For optimal performance all data buffers SHOULD be 8-byte aligned.
为了获得最佳性能，所有数据缓冲区都应 8 字节对齐。

All values in this structure are required to be in Most Significant Byte first order, e.g. a.pData[0] = MSB.
该结构中的所有值都要求以最高有效字节优先，例如 a.pData[0] = MSB。

Note:
注意:

If the client modifies or frees the memory referenced in this structure after it has been submitted to the cpaCyEcdhPointMultiply function, and before it has been returned in the callback, undefined behavior will result.
如果客户端在将此结构中引用的内存提交给 cpaCyEcdhPointMultiply 函数之后，在回调中返回之前修改或释放该内存，将导致未定义的行为。

See also:
另请参见:

cpaCyEcdhPointMultiply()
cpaCyEcdhPointMultiply()

15.5.1 _CpaCyEcdhPointMultiplyOpData Struct Reference

15.5.2_CpaCyEcdhPointMultiplyOpData 结构引用

15.6.2.3 Data Fields

15.6.2.4 数据字段

- **CpaFlatBuffer k**
- CpaFlatBuffer k
- **CpaFlatBuffer xg**
- CpaFlatBuffer xg
- **CpaFlatBuffer yg**
- CpaFlatBuffer yg
- **CpaFlatBuffer a**
- CpaFlatBuffer a
- **CpaFlatBuffer b**
- CpaFlatBuffer b
- **CpaFlatBuffer q**
- CpaFlatBuffer q
- **CpaFlatBuffer h**
- CpaFlatBuffer h
- **CpaCyEcFieldType fieldType**
- CpaCyEcFieldType fieldType
- **CpaBoolean pointVerify**
- CpaBoolean pointVerify

15.6.2.5 Field Documentation
15.6.2.6 现场文件

CpaFlatBuffer _CpaCyEcdhPointMultiplyOpData::k

scalar multiplier (k > 0 and k < n)
标量乘数(k ＞ 0 且 k ＜ n)

CpaFlatBuffer _CpaCyEcdhPointMultiplyOpData::xg

x coordinate of curve point
曲线点的 x 坐标

CpaFlatBuffer _CpaCyEcdhPointMultiplyOpData::yg

y coordinate of curve point
曲线点的 y 坐标

CpaFlatBuffer _CpaCyEcdhPointMultiplyOpData::a

a equation coefficient
方程式系数

CpaFlatBuffer _CpaCyEcdhPointMultiplyOpData::b

b equation coefficient
b 方程系数

CpaFlatBuffer _CpaCyEcdhPointMultiplyOpData::q

prime modulus or irreducible polynomial over GF(2^r)
GF(2^r 上的素数模或不可约多项式)

CpaFlatBuffer _CpaCyEcdhPointMultiplyOpData::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. There are some restrictions on the value of the cofactor. Implementations of this API will support at least the following:

操作的余因子。如果不需要余因子，则将余因子设置为 1 或将平面缓冲区的数据指针设置为空。对于余因子的值有一些限制。该 API 的实现将至少支持以下内容:

- NIST standard curves and their cofactors (1, 2 and 4)
- NIST 标准曲线及其余因子(1、2 和 4)
- Random curves where max(log2(p), log2(n)+log2(h)) <= 512, where p is the modulus, n is the order of the curve and h is the cofactor
- 随机曲线，其中 max(log2(p)，log2(n)+log2(h)) <= 512，其中 p 是模数，n 是曲线的阶，h 是余因子

CpaCyEcFieldType _CpaCyEcdhPointMultiplyOpData::fieldType

CpaCyEcFieldType _CpaCyEcdhPointMultiplyOpData::fieldType

field type for the operation
操作的字段类型

CpaBoolean _CpaCyEcdhPointMultiplyOpData::pointVerify

CpaBoolean _CpaCyEcdhPointMultiplyOpData::pointVerify

set to CPA_TRUE to do a verification before the multiplication
设置 CPA_TRUE 以在乘法之前进行验证

15.5.3 _CpaCyEcdhStats64 Struct Reference

15.5.4 _CpaCyEcdhStats64 结构引用

15.5.2 _CpaCyEcdhStats64 Struct Reference

15.5.3 _CpaCyEcdhStats64 结构引用

15.5.3.1 Detailed Description

15.5.3.2 详细描述

Cryptographic ECDH Statistics.
加密 ECDH 统计。

This structure contains statistics on the Cryptographic ECDH operations. Statistics are set to zero when the component is initialized, and are collected per instance.
该结构包含加密 ECDH 操作的统计信息。当组件初始化时，统计信息被设置为零，并针对每个实例进行收集。

15.5.3.3 Data Fields
15.5.3.4 数据字段

- **Cpa64U numEcdhPointMultiplyRequests**
- Cpa64U numEcdhPointMultiplyRequests
- **Cpa64U numEcdhPointMultiplyRequestErrors**
- Cpa64U numEcdhPointMultiplyRequestErrors
- **Cpa64U numEcdhPointMultiplyCompleted**
- Cpa64U numEcdhPointMultiplyCompleted
- **Cpa64U numEcdhPointMultiplyCompletedError**
- Cpa64U numEcdhPointMultiplyCompletedError
- **Cpa64U numEcdhRequestCompletedOutputInvalid**
- Cpa64U numEcdhRequestCompletedOutputInvalid

15.5.3.5 Field Documentation
15.5.3.6 现场文件

Cpa64U _CpaCyEcdhStats64::numEcdhPointMultiplyRequests
Cpa64U _CpaCyEcdhStats64::numEcdhPointMultiplyRequests

Total number of ECDH Point Multiplication operation requests.
ECDH 点乘操作请求的总数。

Cpa64U _CpaCyEcdhStats64::numEcdhPointMultiplyRequestErrors
Cpa64U _CpaCyEcdhStats64::numEcdhPointMultiplyRequestErrors

Total number of ECDH Point Multiplication operation requests that had an error and could not be processed.
出现错误且无法处理的 ECDH 点乘操作请求的总数。

Cpa64U _CpaCyEcdhStats64::numEcdhPointMultiplyCompleted
Cpa64U _CpaCyEcdhStats64::numEcdhPointMultiplyCompleted

Total number of ECDH Point Multiplication operation requests that completed successfully.
成功完成的 ECDH 点乘操作请求的总数。

Cpa64U _CpaCyEcdhStats64::numEcdhPointMultiplyCompletedError
Cpa64U _CpaCyEcdhStats64::numEcdhPointMultiplyCompletedError

Total number of ECDH Point Multiplication operation requests that could not be completed successfully due to errors.
由于错误而无法成功完成的 ECDH 点乘操作请求的总数。

Cpa64U _CpaCyEcdhStats64::numEcdhRequestCompletedOutputInvalid
Cpa64U _CpaCyEcdhStats64::numEcdhRequestCompletedOutputInvalid

Total number of ECDH Point Multiplication or Point Verify operation requests that could not be completed successfully due to an invalid output. Note that this does not indicate an error.
由于无效输出而无法成功完成的 ECDH 点乘或点验证操作请求的总数。请注意，这并不表示有错误。

15.7 Typedef Documentation

15.8 Typedef 文档

typedef struct _CpaCyEcdhPointMultiplyOpData CpaCyEcdhPointMultiplyOpData

typedef struct _CpaCyEcdhPointMultiplyOpData CpaCyEcdhPointMultiplyOpData

ECDH Point Multiplication Operation Data.
ECDH 点乘运算数据。

This structure contains the operation data for the cpaCyEcdhPointMultiply 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.
此结构包含 cpaCyEcdhPointMultiply 函数的操作数据。客户端必须为这个结构和这个结构指向的项目分配内存。当结构被传递给函数时，内存的所有权就传递给了函数。当这个结构在回调函数中返回时，内存的所有权返回给客户端。

For optimal performance all data buffers SHOULD be 8-byte aligned.
为了获得最佳性能，所有数据缓冲器都应 8 字节对齐。

All values in this structure are required to be in Most Significant Byte first order, e.g. a.pData[0] = MSB.
该结构中的所有值都要求以最高有效字节优先，例如 a.pData[0] = MSB。

Note:
注意:

If the client modifies or frees the memory referenced in this structure after it has been submitted to
如果客户端在将该结构提交给之后修改或释放了该结构中引用的内存

the cpaCyEcdhPointMultiply function, and before it has been returned in the callback, undefined behavior will result.

cpaCyEcdhPointMultiply 函数，在回调中返回之前，将导致未定义的行为。

See also:
另请参见:

cpaCyEcdhPointMultiply()
cpaCyEcdhPointMultiply()

typedef struct **_CpaCyEcdhStats64 CpaCyEcdhStats64**
typedef 结构_CpaCyEcdhStats6CpaCyEcdhStats6

Cryptographic ECDH Statistics.
加密 ECDH 统计。

This structure contains statistics on the Cryptographic ECDH operations. Statistics are set to zero when the component is initialized, and are collected per instance.
该结构包含加密 ECDH 操作的统计信息。当组件初始化时，统计信息被设置为零，并针对每个实例进行收集。

typedef void(* **CpaCyEcdhPointMultiplyCbFunc**)(void *pCallbackTag, **CpaStatus** status, void *pOpD
typedef void(* CpaCyEcdhPointMultiplyCbFunc)(void *pCallbackTag, CpaStatus status, void *pOpData

Definition of callback function invoked for cpaCyEcdhPointMultiply requests.

This is the prototype for the CpaCyEcdhPointMultiplyCbFunc callback function

为 cpaCyEcdhPointMultiply 请求调用的回调函数的定义。这是

CpaCyEcdhPointMultiplyCbFunc 回调函数的原型

Context:
背景:

This callback function can be executed in a context that DOES NOT permit sleeping to occur.
这个回调函数可以在不允许休眠发生的上下文中执行。

Assumptions:
假设:

None
没有人

Side-Effects:
副作用:

None
没有人

Reentrant:
可重入:

No
不

Thread-safe:

线程安全:

Yes

是

Parameters:

参数:

`[in]` *pCallbackTag* User-supplied value to help identify request.

[in] pCallbackTag 使用者提供的值，可协助识别要求。

`[in]` *status* Status of the operation. Valid values are CPA_STATUS_SUCCESS, CPA_STATUS_FAIL and CPA_STATUS_UNSUPPORTED.

[in] status 操作的状态。有效值为 CPA_STATUS_SUCCESS、CPA_STATUS_FAIL 和 CPA_STATUS_UNSUPPORTED。

`[in]` *pOpData* Opaque pointer to Operation data supplied in request.

[in]指向请求中提供的操作数据的 pOpData Opaque 指针。

`[in]` *pXk* Output x coordinate from the request.

[in] pXk 从请求中输出 x 坐标。

`[in]` *pYk* Output y coordinate from the request.

[in] pYk 从请求中输出 y 坐标。

`[in]` *multiplyStatus* Status of the point multiplication and the verification when the pointVerify bit is set in the CpaCyEcdhPointMultiplyOpData structure.

[in]当在 CpaCyEcdhPointMultiplyOpData 结构中设置了 pointVerify 位时，点乘法和验证的 multiplyStatus 状态。

Return values:

返回值:

None

没有人

Precondition:

前提条件:

Component has been initialized.

组件已初始化。

Postcondition:

后置条件:

None

没有人

Note:

注意:

None

没有人

See also:

另请参见:

cpaCyEcdhPointMultiply()

cpaCyEcdhPointMultiply()

15.8 Function Documentation

15.9 功能文档

```
cpaCyEcdhPointMultiply (  const                                          instanceHandle,
cpaCyEcdhPointMultiply (  const                                          instanceHandle,
                               const pCb,
                               void *pCallbackTag,
                               const * pOpData,
                               *pMultiplyStatus,
                               *pXk,
                               *pYk
                              )
```

ECDH Point Multiplication.

ECDH 点乘法。

This function performs ECDH Point Multiplication as defined in ANSI X9.63 2001 section 5.4

该函数执行 ANSI X9.63 2001 第 5.4 节中定义的 ECDH 点乘法

Context:

背景:

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.

当作为异步函数调用时，它不能休眠。它可以在不允许休眠的上下文中执行。当作为同步函数调用时，它可能会休眠。它不能在不允许休眠的上下文中执行。

Assumptions:

假设:

None

没有人

Side-Effects:

副作用:

None

没有人

Blocking:

阻止:

Yes when configured to operate in synchronous mode.

当配置为在同步模式下运行时，是。

Reentrant:
可重入:

No
不

Thread-safe:
线程安全:

Yes
是

Parameters:
参数:

`[in]` *instanceHandle* Instance handle.

[in] instanceHandle 执行个体控制代码。

`[in]` *pCb* Callback function pointer. If this is set to a NULL value the function will operate synchronously.

[in] pCb 回调函数指针。如果设置为空值，函数将同步运行。

`[in]` *pCallbackTag* User-supplied value to help identify request.

[in] pCallbackTag 使用者提供的值，可协助识别要求。

`[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.

[in] pOpData 结构，包含执行作业所需的所有资料。客户端代码为此结构分配内存。该组件取得内存的所有权，直到它在回调中被返回。

`[out]` *pMultiplyStatus* In synchronous mode, the status of the point multiplication and the verification when the pointVerify bit is set in the CpaCyEcdhPointMultiplyOpData structure. 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.

[out] pMultiplyStatus 在同步模式下，当 CpaCyEcdhPointMultiplyOpData 结构中的 pointVerify 位置位时，点乘法的状态和验证。如果点不在曲线上或在无穷远处，则设置为 CPA_FALSE。如果点在曲线上，则设置为 CPA_TRUE。

`[out]` *pXk* Pointer to x coordinate flat buffer.

[out]指向 x 坐标平面缓冲区的 pXk 指针。

`[out]` *pYk* Pointer to y coordinate flat buffer.

[out]指向 y 坐标平面缓冲区的 pYk 指针。

Return values:

返回值:

CPA_STATUS_SUCCESS Function executed successfully.

CPA_STATUS_SUCCESS 函数执行成功。

CPA_STATUS_FAIL Function failed.

CPA_STATUS_FAIL 函数失败。

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.

CPA_STATUS_RETRY 重新提交请求。传递的 CPA_STATUS_INVALID_PARAM 参数无效。与系统资源相关的 CPA_STATUS_RESOURCE 错误。CPA_STATUS_RESTARTING API 实现正在重新启动。重新提交请求。不支持 CPA_STATUS_UNSUPPORTED 函数。

Precondition:

前提条件:

The component has been initialized via cpaCyStartInstance function.

该组件已通过 cpaCyStartInstance 函数初始化。

Postcondition:

后置条件:

None

没有人

Note:

注意:

When pCb is non-NULL an asynchronous callback of type CpaCyEcdhPointMultiplyCbFunc is generated in response to this function call. For optimal performance, data pointers SHOULD be 8-byte aligned.
当pCb为非空时，会生成一个类型为CpaCyEcdhPointMultiplyCbFunc的异步回调来响应此函数调用。为了获得最佳性能，数据指针应该8字节对齐。

See also:

另请参见:

CpaCyEcdhPointMultiplyOpData, **CpaCyEcdhPointMultiplyCbFunc**
CpaCyEcdhPointMultiplyOpData, CpaCyEcdhPointMultiplyCbFunc

CpaStatus cpaCyEcdhQueryStats64 (const **CpaInstanceHandle** *instanceHandle*, **CpaCyEcdhStats64 *** *pEcdhStats*

CpaStatus cpaCyEcdhQueryStats64 (const **CpaInstanceHandle** *instanceHandle*, **CpaCyEcdhStats64 *** *pEcdhStats*

Query statistics for a specific ECDH instance.
查询特定ECDH实例的统计信息。

This function will query a specific instance of the ECDH implementation for statistics. The user MUST allocate the CpaCyEcdhStats64 structure and pass the reference to that structure into this function call. This function writes the statistic results into the passed in CpaCyEcdhStats64 structure.
该函数将查询ECDH实现的特定实例的统计信息。用户必须分配CpaCyEcdhStats64结构，并将对该结构的引用传递到此函数调用中。此函数将统计结果写入传入的CpaCyEcdhStats64结构中。

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.
注意:此函数返回的统计数据不会中断当前的数据处理，因此可能会与统计数据检索过程中正在进行的操作稍微不同步。

Context:
背景:

This is a synchronous function and it can sleep. It MUST NOT be executed in a context that DOES NOT permit sleeping.
这是一个同步功能，它可以休眠。它不能在不允许休眠的上下文中执行。

Assumptions:
假设:

None
没有人

Side-Effects:
副作用:

None
没有人

Blocking:
阻止:

This function is synchronous and blocking.
这个函数是同步的和阻塞的。

Reentrant:
可重入:

No
不

Thread-safe:
线程安全:
Yes
是

Parameters:
参数:

`[in]` *instanceHandle* Instance handle.

[in] instanceHandle 执行个体控制代码。

`[out]` *pEcdhStats* Pointer to memory into which the statistics will be written.

[out]指向将写入统计信息的内存的 pEcdhStats 指针。

Return values:

返回值:

CPA_STATUS_SUCCESS Function executed successfully.

CPA_STATUS_SUCCESS 函数执行成功。

CPA_STATUS_FAIL Function failed.

CPA_STATUS_INVALID_PARAM Invalid parameter passed in.

CPA_STATUS_RESOURCE Error related to system resources.

CPA_STATUS_FAIL 函数失败。传递的 CPA_STATUS_INVALID_PARAM 参数无效。与系统资源相关的 CPA_STATUS_RESOURCE 错误。

CPA_STATUS_RESTARTING API implementation is restarting. Resubmit the request.

CPA_STATUS_RESTARTING API 实现正在重新启动。重新提交请求。

CPA_STATUS_UNSUPPORTED Function is not supported.

不支持 CPA_STATUS_UNSUPPORTED 函数。

Precondition:

前提条件:

Component has been initialized.

组件已初始化。

Postcondition:

后置条件:

None

没有人

Note:

注意:

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

该函数以同步方式运行，不会生成异步回调。

See also:

另请参见:

CpaCyEcdhStats64

CpaCyEcdhStats64

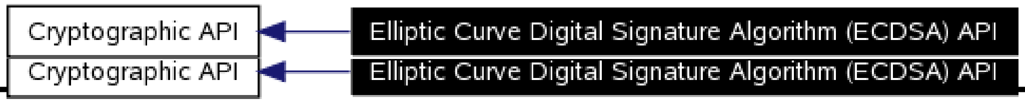
18 Elliptic Curve Digital Signature Algorithm (ECDSA) API

19 椭圆曲线数字签名算法(ECDSA) API

[Cryptographic API]

[Cryptographic API]

Collaboration diagram for Elliptic Curve Digital Signature Algorithm (ECDSA) API:
椭圆曲线数字签名算法(ECDSA) API 的协作图:


Cryptographic API
Elliptic Curve Digital Signature Algorithm (ECDSA) API
Cryptographic API
Elliptic Curve Digital Signature Algorithm (ECDSA) API

19.1 Detailed Description

19.2 详细描述

File: cpa_cy_ecdsa.h

文件:cpa_cy_ecdsa.h

These functions specify the API for Public Key Encryption (Cryptography) Elliptic Curve Digital Signature Algorithm (ECDSA) operations.
这些函数指定了用于公钥加密(加密)椭圆曲线数字签名算法(ECDSA)操作的 API。

Note:
注意:

Large numbers are represented on the QuickAssist API as described in the Large Number API (**Cryptographic Large Number API**).
大数在 QuickAssist API 上表示，如大数 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.
此外，对于椭圆曲线运算，传递给 API 的大数的位长度不得超过 576 位。

19.3 Data Structures

19.4 数据结构

- struct **_CpaCyEcdsaSignROpData**

- 结构体_CpaCyEcdsaSignROpData

- struct **_CpaCyEcdsaSignSOpData**
- 结构体_CpaCyEcdsaSignSOpData
- struct **_CpaCyEcdsaSignRSOpData**
- 结构体_CpaCyEcdsaSignRSOpData
- struct **_CpaCyEcdsaVerifyOpData**
- 结构体_CpaCyEcdsaVerifyOpData
- struct **_CpaCyEcdsaStats64**
- 结构体_CpaCyEcdsaStats64

19.5 Typedefs

19.6 类型定义

- typedef **_CpaCyEcdsaSignROpData CpaCyEcdsaSignROpData**

- 数据类型说明_CpaCyEcdsaSignROpData CpaCyEcdsaSignROpData
- typedef **_CpaCyEcdsaSignSOpData CpaCyEcdsaSignSOpData**
- 数据类型说明_CpaCyEcdsaSignSOpData CpaCyEcdsaSignSOpData
- typedef **_CpaCyEcdsaSignRSOpData CpaCyEcdsaSignRSOpData**
- 数据类型说明_CpaCyEcdsaSignRSOpData CpaCyEcdsaSignRSOpData
- typedef **_CpaCyEcdsaVerifyOpData CpaCyEcdsaVerifyOpData**
- 数据类型说明_CpaCyEcdsaVerifyOpData CpaCyEcdsaVerifyOpData
- typedef **_CpaCyEcdsaStats64 CpaCyEcdsaStats64**
- 数据类型说明_CpaCyEcdsaStats64 CpaCyEcdsaStats64
- typedef void(* **CpaCyEcdsaGenSignCbFunc**)(void *pCallbackTag, **CpaStatus** status, void
- typedef void(*CpaCyEcdsaGenSignCbFunc CpaStatus
  *pOpData, **CpaBoolean** multiplyStatus, **CpaFlatBuffer** *pOut)
  *pOpData，CpaBoolean CpaFlatBuffer
- typedef void(* **CpaCyEcdsaSignRSCbFunc**)(void *pCallbackTag, **CpaStatus** status, void
- typedef void(*CpaCyEcdsaSignRSCbFunc CpaStatus
  *pOpData, **CpaBoolean** multiplyStatus, **CpaFlatBuffer** *pR, **CpaFlatBuffer** *pS)
  *pOpData，CpaBoolean CpaFlatBuffer CpaFlatBuffer
- typedef void(* **CpaCyEcdsaVerifyCbFunc**)(void *pCallbackTag, **CpaStatus** status, void *pOpData,
- typedef void(*CpaCyEcdsaVerifyCbFunc CpaStatus
  CpaBoolean verifyStatus)
  CpaBoolean 验证状态)

19.7 Functions

19.8 功能

- **CpaStatus cpaCyEcdsaSignR** (const **CpaInstanceHandle** instanceHandle, const

- CpaStatus cpaCyEcdsaSignR（常量CpaInstanceHandle
  CpaCyEcdsaGenSignCbFunc pCb, void *pCallbackTag, const **CpaCyEcdsaSignROpData**
  CpaCyEcdsaGenSignCbFunc pCb，void *pCallbackTag，constCpaCyEcdsaSignROpData
  *pOpData, **CpaBoolean** *pSignStatus, **CpaFlatBuffer** *pR)
  *pOpData，CpaBoolean CpaFlatBuffer
- **CpaStatus cpaCyEcdsaSignS** (const **CpaInstanceHandle** instanceHandle, const
- CpaStatus cpaCyEcdsaSignS（常量CpaInstanceHandle

16.4 Functions

16.5 功能

CpaCyEcdsaGenSignCbFunc pCb, void *pCallbackTag, const **CpaCyEcdsaSignSOpData**

CpaCyEcdsaGenSignCbFunc pCb，void *pCallbackTag，constCpaCyEcdsaSignSOpData
*pOpData, **CpaBoolean** *pSignStatus, **CpaFlatBuffer** *pS)
*pOpData，CpaBoolean CpaFlatBuffer

- **CpaStatus cpaCyEcdsaSignRS** (const **CpaInstanceHandle** instanceHandle, const
- CpaStatus cpaCyEcdsaSignRS（常量CpaInstanceHandle
CpaCyEcdsaSignRSCbFunc pCb, void *pCallbackTag, const **CpaCyEcdsaSignRSOpData**
CpaCyEcdsaSignRSCbFunc pCb，void *pCallbackTag，constCpaCyEcdsaSignRSOpData
*pOpData, **CpaBoolean** *pSignStatus, **CpaFlatBuffer** *pR, **CpaFlatBuffer** *pS)
*pOpData，CpaBoolean CpaFlatBuffer CpaFlatBuffer
- **CpaStatus cpaCyEcdsaVerify** (const **CpaInstanceHandle** instanceHandle, const **CpaCyEcdsaVerifyCbFunc** pCb, void *pCallbackTag, const **CpaCyEcdsaVerifyOpData** *pOpData, **CpaBoolean** *pVerifyStatus)
- CpaStatus cpaCyEcdsaVerify（常量CpaInstanceHandle CpaCyEcdsaVerifyCbFunc CpaCyEcdsaVerifyOpData CpaBoolean
- **CpaStatus cpaCyEcdsaQueryStats64** (const **CpaInstanceHandle** instanceHandle,
- CpaStatus cpaCyEcdsaQueryStats64（常量CpaInstanceHandle
CpaCyEcdsaStats64 *pEcdsaStats)
CpaCyEcdsaStats64 * pEcdsaStats)

16.6 Data Structure Documentation

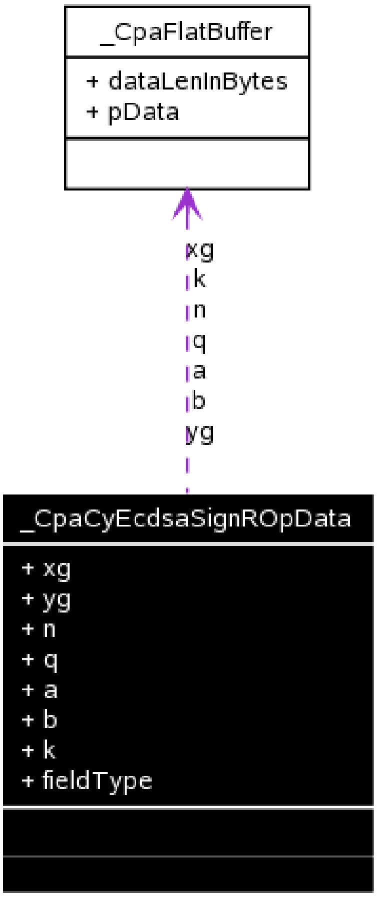
16.7 数据结构文档

16.7.1 _CpaCyEcdsaSignROpData Struct Reference

16.7.2 _ CpaCyEcdsaSignROpData结构引用

Collaboration diagram for _CpaCyEcdsaSignROpData:

_ CpaCyEcdsaSignROpData的协作图:



16.7.2.1 Detailed Description
16.7.2.2 详细描述

ECDSA Sign R Operation Data.
ECDSA 标志 R 操作数据。

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.
此结构包含 cpaCyEcdsaSignR 函数的操作数据。客户端必须为这个结构和这个结构指向的项目分配内存。当结构被传递给函数时，内存的所有权就传递给了函数。当这个结构在回调函数中返回时，内存的所有权返回给客户端。

16.5.1 _CpaCyEcdsaSignROpData Struct Reference

16.5.2_ CpaCyEcdsaSignROpData 结构引用

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

为了获得最佳性能，所有数据缓冲器都应8字节对齐。

All values in this structure are required to be in Most Significant Byte first order, e.g. a.pData[0] = MSB.
该结构中的所有值都要求以最高有效字节优先，例如a.pData[0] = MSB。

Note:
注意:

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.
如果客户端在将此结构中引用的内存提交给cpaCyEcdsaSignR函数之后，在回调中返回之前修改或释放该内存，将导致未定义的行为。

See also:
另请参见:

cpaCyEcdsaSignR()
cpaCyEcdsaSignR()

16.7.2.3 Data Fields
16.7.2.4 数据字段

- **CpaFlatBuffer xg**
- CpaFlatBuffer xg
- **CpaFlatBuffer yg**
- CpaFlatBuffer yg
- **CpaFlatBuffer n**
- CpaFlatBuffer n
- **CpaFlatBuffer q**
- CpaFlatBuffer q
- **CpaFlatBuffer a**
- CpaFlatBuffer a
- **CpaFlatBuffer b**
- CpaFlatBuffer b
- **CpaFlatBuffer k**
- CpaFlatBuffer k
- **CpaCyEcFieldType fieldType**
- CpaCyEcFieldType fieldType

16.7.2.5 Field Documentation
16.7.2.6 现场文件

CpaFlatBuffer _CpaCyEcdsaSignROpData::xg
CpaFlatBuffer CpaCyEcdsaSignROpData

x coordinate of base point G
基点G的x坐标

CpaFlatBuffer _CpaCyEcdsaSignROpData::yg
CpaFlatBuffer CpaCyEcdsaSignROpData

y coordinate of base point G

基点G的y坐标

CpaFlatBuffer CpaCyEcdsaSignROpData::n

order of the base point G, which shall be prime
应该是质数的基点G的阶

CpaFlatBuffer CpaCyEcdsaSignROpData::q

prime modulus or irreducible polynomial over GF(2^r)
GF(2^r上的素数模或不可约多项式)

CpaFlatBuffer CpaCyEcdsaSignROpData::a

a elliptic curve coefficient
椭圆曲线系数

CpaFlatBuffer CpaCyEcdsaSignROpData::b

b elliptic curve coefficient
b椭圆曲线系数

CpaFlatBuffer CpaCyEcdsaSignROpData::k

random value (k > 0 and k < n)
随机值(k ＞ 0且k ＜ n)

CpaCyEcFieldType CpaCyEcdsaSignROpData::fieldType

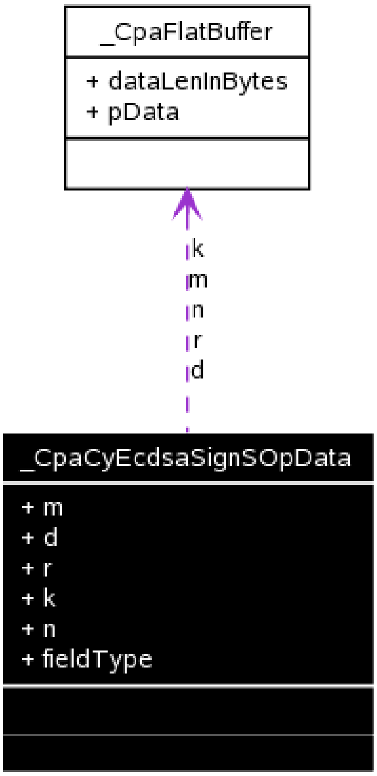
field type for the operation
操作的字段类型

16.5.3 _CpaCyEcdsaSignSOpData Struct Reference

16.5.4 _ CpaCyEcdsaSignSOpData 结构引用

Collaboration diagram for _CpaCyEcdsaSignSOpData:

_ CpaCyEcdsaSignSOpData 的协作图:



16.5.4.1 Detailed Description

16.5.4.2 详细描述

ECDSA Sign S Operation Data.
ECDSA 标志的操作数据。

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.
此结构包含 cpaCyEcdsaSignS 函数的操作数据。客户端必须为这个结构和这个结构指向的项目分配内存。当结构被传递给函数时，内存的所有权就传递给了函数。当这个结构在回调函数中返回时，内存的所有权返回给客户端。

For optimal performance all data buffers SHOULD be 8-byte aligned.
为了获得最佳性能，所有数据缓冲器都应 8 字节对齐。

All values in this structure are required to be in Most Significant Byte first order, e.g. a.pData[0] = MSB.
该结构中的所有值都要求以最高有效字节优先，例如 a.pData[0] = MSB。

Note:
注意:

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.
如果客户端在将此结构中引用的内存提交给 cpaCyEcdsaSignS 函数之后，在回调中返回之前修改或释放该内存，将导致未定义的行为。

See also:
另请参见:

cpaCyEcdsaSignS()
cpaCyEcdsaSignS()

16.5.4.3 Data Fields
16.5.4.4 数据字段

- **CpaFlatBuffer m**
- CpaFlatBuffer m
- **CpaFlatBuffer d**
- CpaFlatBuffer d
- **CpaFlatBuffer r**
- CpaFlatBuffer r

- **CpaFlatBuffer k**
- CpaFlatBuffer k
- **CpaFlatBuffer n**
- CpaFlatBuffer n
- **CpaCyEcFieldType fieldType**
- CpaCyEcFieldType fieldType

16.5.4.5 Field Documentation

16.5.4.6 现场文件

CpaFlatBuffer _CpaCyEcdsaSignSOpData::m

CpaFlatBuffer _CpaCyEcdsaSignSOpData::m

digest of the message to be signed

要签名的消息的摘要

CpaFlatBuffer _CpaCyEcdsaSignSOpData::d

CpaFlatBuffer _CpaCyEcdsaSignSOpData::d

private key

私人密钥

CpaFlatBuffer _CpaCyEcdsaSignSOpData::r

CpaFlatBuffer _CpaCyEcdsaSignSOpData::r

Ecdsa r signature value

Ecdsa r 签名值

CpaFlatBuffer _CpaCyEcdsaSignSOpData::k

CpaFlatBuffer _CpaCyEcdsaSignSOpData::k

random value (k > 0 and k < n)

随机值(k > 0 且 k < n)

CpaFlatBuffer _CpaCyEcdsaSignSOpData::n

CpaFlatBuffer _CpaCyEcdsaSignSOpData::n

order of the base point G, which shall be prime

应该是质数的基点 G 的阶

CpaCyEcFieldType _CpaCyEcdsaSignSOpData::fieldType

CpaCyEcFieldType _CpaCyEcdsaSignSOpData::fieldType

field type for the operation

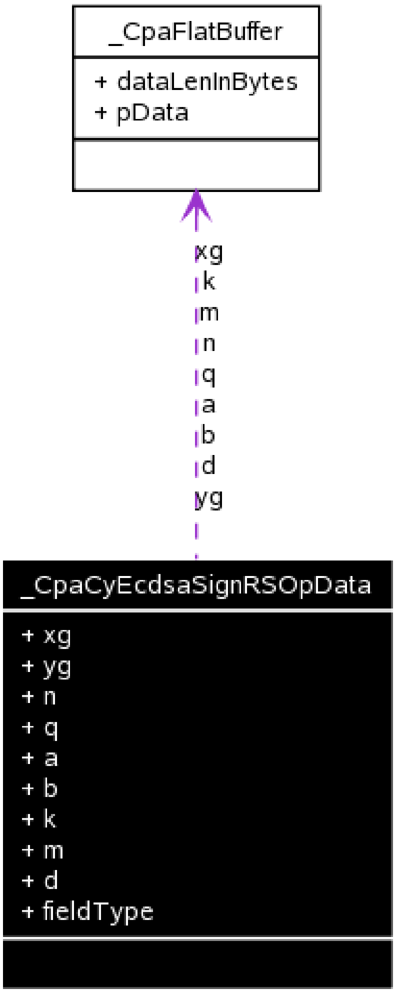
操作的字段类型

16.5.5 _CpaCyEcdsaSignRSOpData Struct Reference

16.5.6 _ CpaCyEcdsaSignRSOpData 结构引用

Collaboration diagram for _CpaCyEcdsaSignRSOpData:

_ CpaCyEcdsaSignRSOpData 的协作图:



16.5.6.1 Detailed Description

16.5.6.2 详细描述

ECDSA Sign R & S Operation Data.
R & S 运营数据。

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.
此结构包含 cpaCyEcdsaSignRS 函数的操作数据。客户端必须为这个结构和这个结构指向的项目分配内存。当结构被传递给函数时，内存的所有权就传递给了函数。当这个结构在回调函数中返回时，内存的所有权返回给客户端。

For optimal performance all data buffers SHOULD be 8-byte aligned.
为了获得最佳性能，所有数据缓冲器都应 8 字节对齐。

All values in this structure are required to be in Most Significant Byte first order, e.g. a.pData[0] = MSB.
该结构中的所有值都要求以最高有效字节优先，例如 a.pData[0] = MSB。

Note:
注意:

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.
如果客户端在将此结构中引用的内存提交给 cpaCyEcdsaSignRS 函数之后，在回调中返回之前修改或释放该内存，将导致未定义的行为。

See also:
另请参见:

cpaCyEcdsaSignRS()
cpaCyEcdsaSignRS()

16.5.6.3 Data Fields

16.5.6.4 数据字段

- **CpaFlatBuffer xg**
- CpaFlatBuffer xg
- **CpaFlatBuffer yg**
- CpaFlatBuffer yg
- **CpaFlatBuffer n**
- CpaFlatBuffer n
- **CpaFlatBuffer q**
- CpaFlatBuffer q
- **CpaFlatBuffer a**
- CpaFlatBuffer a
- **CpaFlatBuffer b**
- CpaFlatBuffer b
- **CpaFlatBuffer k**
- CpaFlatBuffer k
- **CpaFlatBuffer m**
- CpaFlatBuffer m
- **CpaFlatBuffer d**
- CpaFlatBuffer d
- **CpaCyEcFieldType fieldType**
- CpaCyEcFieldType fieldType

16.5.6.5 Field Documentation

16.5.6.6 现场文件

CpaFlatBuffer CpaCyEcdsaSignRSOpData::xg

x coordinate of base point G

基点 G 的 x 坐标

CpaFlatBuffer CpaCyEcdsaSignRSOpData::yg

y coordinate of base point G

基点 G 的 y 坐标

CpaFlatBuffer CpaCyEcdsaSignRSOpData::n

order of the base point G, which shall be prime

应该是质数的基点 G 的阶

CpaFlatBuffer CpaCyEcdsaSignRSOpData::q

prime modulus or irreducible polynomial over GF(2^r)

GF(2^r 上的素数模或不可约多项式)

CpaFlatBuffer CpaCyEcdsaSignRSOpData::a

a elliptic curve coefficient

椭圆曲线系数

CpaFlatBuffer CpaCyEcdsaSignRSOpData::b

b elliptic curve coefficient

b 椭圆曲线系数

CpaFlatBuffer CpaCyEcdsaSignRSOpData::k

random value (k > 0 and k < n)

随机值(k > 0 且 k < n)

CpaFlatBuffer* CpaCyEcdsaSignRSOpData::m

CpaFlatBuffer* CpaCyEcdsaSignRSOpData::m

digest of the message to be signed
要签名的消息的摘要

CpaFlatBuffer* CpaCyEcdsaSignRSOpData::d

CpaFlatBuffer* CpaCyEcdsaSignRSOpData::d

private key
私人密钥

CpaCyEcFieldType CpaCyEcdsaSignRSOpData::fieldType

CpaCyEcFieldType CpaCyEcdsaSignRSOpData::fieldType

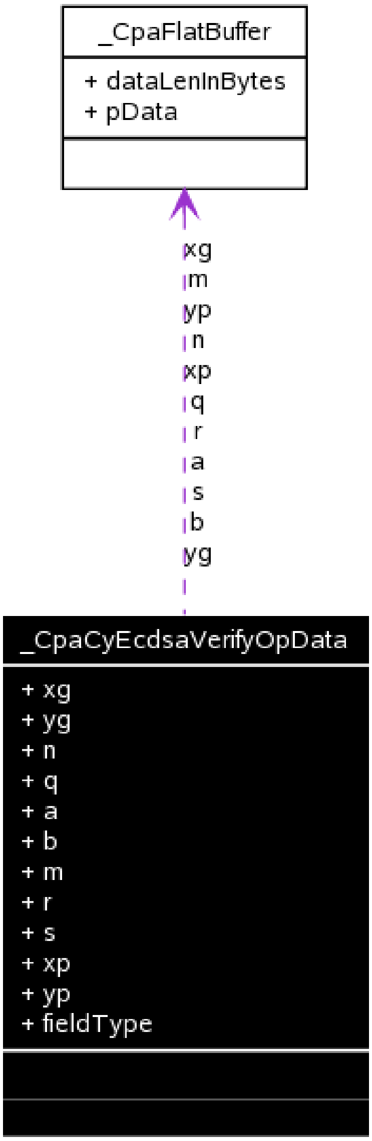
field type for the operation
操作的字段类型

16.5.7 _CpaCyEcdsaVerifyOpData Struct Reference

16.5.8 _CpaCyEcdsaVerifyOpData 结构引用

Collaboration diagram for _CpaCyEcdsaVerifyOpData:

_CpaCyEcdsaVerifyOpData 的协作图:



16.5.8.1 Detailed Description

16.5.8.2 详细描述

ECDSA Verify Operation Data, for Public Key.
ECDSA 验证公共密钥的操作数据。

16.5.4 CpaCyEcdsaVerifyOpData Struct

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.
此结构包含CpaCyEcdsaVerify函数的操作数据。客户端必须为这个结构和这个结构指向的项目分配内存。当结构被传递给函数时，内存的所有权就传递给了函数。当这个结构在回调函数中返回时，内存的所有权返回给客户端。

For optimal performance all data buffers SHOULD be 8-byte aligned.
为了获得最佳性能，所有数据缓冲器都应8字节对齐。

All values in this structure are required to be in Most Significant Byte first order, e.g. a.pData[0] = MSB.
该结构中的所有值都要求以最高有效字节优先，例如a.pData[0] = MSB。

Note:
注意:

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.

如果客户端在将此结构中引用的内存提交给 cpaCyEcdsaVerify 函数之后，在回调中返回之前修改或释放该内存，将导致未定义的行为。

See also:
另请参见:

CpaCyEcdsaVerify()
CpaCyEcdsaVerify()

16.5.8.3 Data Fields
16.5.8.4 数据字段

- **CpaFlatBuffer xg**
- CpaFlatBuffer xg
- **CpaFlatBuffer yg**
- CpaFlatBuffer yg
- **CpaFlatBuffer n**
- CpaFlatBuffer n
- **CpaFlatBuffer q**
- CpaFlatBuffer q
- **CpaFlatBuffer a**
- CpaFlatBuffer a
- **CpaFlatBuffer b**
- CpaFlatBuffer b
- **CpaFlatBuffer m**
- CpaFlatBuffer m
- **CpaFlatBuffer r**
- CpaFlatBuffer r
- **CpaFlatBuffer s**
- CpaFlatBuffer s
- **CpaFlatBuffer xp**
- CpaFlatBuffer xp
- **CpaFlatBuffer yp**
- CpaFlatBuffer yp
- **CpaCyEcFieldType fieldType**
- CpaCyEcFieldType fieldType

16.5.8.5 Field Documentation
16.5.8.6 现场文件

CpaFlatBuffer CpaCyEcdsaVerifyOpData::xg
CpaFlatBuffer CpaCyEcdsaVerifyOpDat

x coordinate of base point G
基点 G 的 x 坐标

CpaFlatBuffer CpaCyEcdsaVerifyOpData::yg
CpaFlatBuffer CpaCyEcdsaVerifyOpDat

y coordinate of base point G
基点 G 的 y 坐标

CpaFlatBuffer CpaCyEcdsaVerifyOpData::n
CpaFlatBuffer CpaCyEcdsaVerifyOpDat

order of the base point G, which shall be prime

应该是质数的基点 G 的阶

CpaFlatBuffer CpaCyEcdsaVerifyOpData::q

prime modulus or irreducible polynomial over GF(2^r)

GF(2^r 上的素数模或不可约多项式)

CpaFlatBuffer CpaCyEcdsaVerifyOpData::a

a elliptic curve coefficient

椭圆曲线系数

CpaFlatBuffer CpaCyEcdsaVerifyOpData::b

b elliptic curve coefficient

b 椭圆曲线系数

CpaFlatBuffer CpaCyEcdsaVerifyOpData::m

digest of the message to be signed

要签名的消息的摘要

CpaFlatBuffer CpaCyEcdsaVerifyOpData::r

ECDSA r signature value (r > 0 and r < n)

ECDSA r 签名值(r > 0 且 r < n)

CpaFlatBuffer CpaCyEcdsaVerifyOpData::s

ECDSA s signature value (s > 0 and s < n)

ECDSA 的签名值(s > 0 且 s < n)

CpaFlatBuffer CpaCyEcdsaVerifyOpData::xp

CpaFlatBuffer CpaCyEcdsaVerifyOpData::xp

x coordinate of point P (public key)
点 P 的 x 坐标(公钥)

CpaFlatBuffer CpaCyEcdsaVerifyOpData::yp

CpaFlatBuffer CpaCyEcdsaVerifyOpData::yp

y coordinate of point P (public key)
点 P 的 y 坐标(公钥)

CpaCyEcFieldType CpaCyEcdsaVerifyOpData::fieldType

CpaCyEcFieldType CpaCyEcdsaVerifyOpData::fieldType

field type for the operation
操作的字段类型

16.5.9 _CpaCyEcdsaStats64 Struct Reference

16.5.10 _ CpaCyEcdsaStats64 结构引用

16.5.10.1 Detailed Description

16.5.10.2 详细描述

Cryptographic ECDSA Statistics.
加密 ECDSA 统计。

This structure contains statistics on the Cryptographic ECDSA operations. Statistics are set to zero when the component is initialized, and are collected per instance.
此结构包含有关加密 ECDSA 操作的统计信息。当组件初始化时，统计信息被设置为零，并针对每个实例进行收集。

16.5.10.3 Data Fields

16.5.10.4 数据字段

- **Cpa64U numEcdsaSignRRequests**
- Cpa64U numEcdsaSignRRequests
- **Cpa64U numEcdsaSignRRequestErrors**
- Cpa64U numEcdsaSignRRequestErrors
- **Cpa64U numEcdsaSignRCompleted**
- Cpa64U numEcdsaSignRCompleted
- **Cpa64U numEcdsaSignRCompletedErrors**
- Cpa64U numEcdsaSignRCompletedErrors
- **Cpa64U numEcdsaSignRCompletedOutputInvalid**
- Cpa64U numEcdsaSignRCompletedOutputInvalid
- **Cpa64U numEcdsaSignSRequests**
- Cpa64U numEcdsaSignSRequests
- **Cpa64U numEcdsaSignSRequestErrors**
- Cpa64U numEcdsaSignSRequestErrors
- **Cpa64U numEcdsaSignSCompleted**
- Cpa64U numEcdsaSignSCompleted
- **Cpa64U numEcdsaSignSCompletedErrors**
- Cpa64U numEcdsaSignSCompletedErrors

- **Cpa64U numEcdsaSignSCompletedOutputInvalid**
- Cpa64U numEcdsaSignSCompletedOutputInvalid
- **Cpa64U numEcdsaSignRSRequests**
- Cpa64U numEcdsaSignRSRequests
- **Cpa64U numEcdsaSignRSRequestErrors**
- Cpa64U numEcdsaSignRSRequestErrors
- **Cpa64U numEcdsaSignRSCompleted**
- Cpa64U numEcdsaSignRSCompleted
- **Cpa64U numEcdsaSignRSCompletedErrors**
- Cpa64U numEcdsaSignRSCompletedErrors
- **Cpa64U numEcdsaSignRSCompletedOutputInvalid**
- Cpa64U numEcdsaSignRSCompletedOutputInvalid
- **Cpa64U numEcdsaVerifyRequests**
- Cpa64U numEcdsaVerifyRequests
- **Cpa64U numEcdsaVerifyRequestErrors**
- Cpa64U numEcdsaVerifyRequestErrors
- **Cpa64U numEcdsaVerifyCompleted**
- Cpa64U numEcdsaVerifyCompleted
- **Cpa64U numEcdsaVerifyCompletedErrors**
- Cpa64U numEcdsaVerifyCompletedErrors
- **Cpa64U numEcdsaVerifyCompletedOutputInvalid**
- Cpa64U numEcdsaVerifyCompletedOutputInvalid

16.5.10.5 Field Documentation

16.5.10.6 现场文件

Cpa64U CpaCyEcdsaStats64::numEcdsaSignRRequests

Total number of ECDSA Sign R operation requests.

ECDSA Sign R 操作请求的总数。

Cpa64U CpaCyEcdsaStats64::numEcdsaSignRRequestErrors

Total number of ECDSA Sign R operation requests that had an error and could not be processed.

出现错误且无法处理的ECDSA Sign R 操作请求的总数。

Cpa64U CpaCyEcdsaStats64::numEcdsaSignRCompleted

Total number of ECDSA Sign R operation requests that completed successfully.

成功完成的ECDSA Sign R 操作请求的总数。

Cpa64U CpaCyEcdsaStats64::numEcdsaSignRCompletedErrors

Total number of ECDSA Sign R operation requests that could not be completed successfully due to errors.
由于错误而无法成功完成的ECDSA Sign R操作请求的总数。

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.
由于无效输出而无法成功完成的ECDSA Sign R操作请求总数。请注意，这并不表示有错误。

Cpa64U CpaCyEcdsaStats64::numEcdsaSignSRequests

Total number of ECDSA Sign S operation requests.
ECDSA签名操作请求的总数。

Cpa64U CpaCyEcdsaStats64::numEcdsaSignSRequestErrors

Total number of ECDSA Sign S operation requests that had an error and could not be processed.
出现错误且无法处理的ECDSA签名操作请求的总数。

Cpa64U CpaCyEcdsaStats64::numEcdsaSignSCompleted

Total number of ECDSA Sign S operation requests that completed successfully.
成功完成的ECDSA签名操作请求的总数。

Cpa64U CpaCyEcdsaStats64::numEcdsaSignSCompletedErrors

Total number of ECDSA Sign S operation requests that could not be completed successfully due to errors.
由于错误而无法成功完成的ECDSA签名操作请求的总数。

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.
由于无效输出而无法成功完成的ECDSA签名操作请求总数。请注意，这并不表示有错误。

Cpa64U CpaCyEcdsaStats64::numEcdsaSignRSRequests

Total number of ECDSA Sign R & S operation requests.
ECDSA签名R & S操作请求的总数。

Cpa64U CpaCyEcdsaStats64::numEcdsaSignRSRequestErrors

Total number of ECDSA Sign R & S operation requests that had an error and could not be processed.
出现错误且无法处理的ECDSA签名R & S操作请求的总数。

Cpa64U CpaCyEcdsaStats64::numEcdsaSignRSCompleted

Total number of ECDSA Sign R & S operation requests that completed successfully.
成功完成的ECDSA签名R & S操作请求的总数。

Cpa64U CpaCyEcdsaStats64::numEcdsaSignRSCompletedErrors

Total number of ECDSA Sign R & S operation requests that could not be completed successfully due to errors.
由于错误而无法成功完成的ECDSA签名R & S操作请求的总数。

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.
由于输出无效而无法成功完成的ECDSA签名R & S操作请求总数。请注意，这并不表示有错误。

Cpa64U CpaCyEcdsaStats64::numEcdsaVerifyRequests

Total number of ECDSA Verification operation requests.
ECDSA验证操作请求的总数。

Cpa64U CpaCyEcdsaStats64::numEcdsaVerifyRequestErrors

Total number of ECDSA Verification operation requests that had an error and could not be processed.

出现错误且无法处理的ECDSA验证操作请求的总数。

Cpa64U CpaCyEcdsaStats64::numEcdsaVerifyCompleted

Total number of ECDSA Verification operation requests that completed successfully.

Cpa64U CpaCyEcdsaStats64::numEcdsaVerifyCompleted

成功完成的ECDSA验证操作请求的总数。

Cpa64U CpaCyEcdsaStats64::numEcdsaVerifyCompletedErrors

Cpa64U CpaCyEcdsaStats64::numEcdsaVerifyCompletedErrors

Total number of ECDSA Verification operation requests that could not be completed successfully due to errors.

由于错误而无法成功完成的 ECDSA 验证操作请求的总数。

Cpa64U CpaCyEcdsaStats64::numEcdsaVerifyCompletedOutputInvalid

Total number of ECDSA Verification operation requests that resulted in an invalid output. Note that this does not indicate an error.

导致无效输出的 ECDSA 验证操作请求的总数。请注意，这并不表示有错误。

16.8 Typedef Documentation

16.9 Typedef 文档

typedef struct **_CpaCyEcdsaSignROpData CpaCyEcdsaSignROpData**

ECDSA Sign R Operation Data.

ECDSA 标志 R 操作数据。

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.

此结构包含 cpaCyEcdsaSignR 函数的操作数据。客户端必须为这个结构和这个结构指向的项目分配内存。当结构被传递给函数时，内存的所有权就传递给了函数。当这个结构在回调函数中返回时，内存的所有权返回给客户端。

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

为了获得最佳性能，所有数据缓冲器都应 8 字节对齐。

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

该结构中的所有值都要求以最高有效字节优先，例如 a.pData[0] = MSB。

Note:

注意:

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.

如果客户端在将此结构中引用的内存提交给 cpaCyEcdsaSignR 函数之后，在回调中返回之前修改或释放该内存，将导致未定义的行为。

See also:

另请参见:

cpaCyEcdsaSignR()

cpaCyEcdsaSignR()

typedef struct **_CpaCyEcdsaSignSOpData CpaCyEcdsaSignSOpData**

ECDSA Sign S Operation Data.

ECDSA 标志的操作数据。

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.
此结构包含 cpaCyEcdsaSignS 函数的操作数据。客户端必须为这个结构和这个结构指向的项目分配内存。当结构被传递给函数时，内存的所有权就传递给了函数。当这个结构在回调函数中返回时，内存的所有权返回给客户端。

For optimal performance all data buffers SHOULD be 8-byte aligned.
为了获得最佳性能，所有数据缓冲器都应 8 字节对齐。

All values in this structure are required to be in Most Significant Byte first order, e.g. a.pData[0] = MSB.
该结构中的所有值都要求以最高有效字节优先，例如 a.pData[0] = MSB。

Note:
注意：

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.
如果客户端在将此结构中引用的内存提交给 cpaCyEcdsaSignS 函数之后，在回调中返回之前修改或释放该内存，将导致未定义的行为。

See also:
另请参见：

cpaCyEcdsaSignS()
cpaCyEcdsaSignS()

typedef struct _CpaCyEcdsaSignRSOpData CpaCyEcdsaSignRSOpData

typedef struct _CpaCyEcdsaSignRSOpData CpaCyEcdsaSignRSOpData

ECDSA Sign R & S Operation Data.
R & S 运营数据。

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.

此结构包含 cpaCyEcdsaSignRS 函数的操作数据。客户端必须为这个结构和这个结构指向的项目分配内存。当结构被传递给函数时，内存的所有权就传递给了函数。当这个结构在回调函数中返回时，内存的所有权返回给客户端。

For optimal performance all data buffers SHOULD be 8-byte aligned.
为了获得最佳性能，所有数据缓冲器都应 8 字节对齐。

All values in this structure are required to be in Most Significant Byte first order, e.g. a.pData[0] = MSB.
该结构中的所有值都要求以最高有效字节优先，例如 a.pData[0] = MSB。

Note:
注意:

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.
如果客户端在将此结构中引用的内存提交给 cpaCyEcdsaSignRS 函数之后，在回调中返回之前修改或释放该内存，将导致未定义的行为。

See also:
另请参见:

cpaCyEcdsaSignRS()
cpaCyEcdsaSignRS()

typedef struct **_CpaCyEcdsaVerifyOpData CpaCyEcdsaVerifyOpData**
typedef 结构**_CpaCyEcdsaVerifyOpDatCpaCyEcdsaVerifyOpDat**

ECDSA Verify Operation Data, for Public Key.
ECDSA 验证公共密钥的操作数据。

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.
此结构包含 CpaCyEcdsaVerify 函数的操作数据。客户端必须为这个结构和这个结构指向的项目分配内存。当结构被传递给函数时，内存的所有权就传递给了函数。当这个结构在回调函数中返回时，内存的所有权返回给客户端。

For optimal performance all data buffers SHOULD be 8-byte aligned.
为了获得最佳性能，所有数据缓冲器都应 8 字节对齐。

All values in this structure are required to be in Most Significant Byte first order, e.g. a.pData[0] = MSB.
该结构中的所有值都要求以最高有效字节优先，例如 a.pData[0] = MSB。

Note:

注意:

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.
如果客户端在将此结构中引用的内存提交给 cpaCyEcdsaVerify 函数之后，在回调中返回之前修改或释放该内存，将导致未定义的行为。

See also:
另请参见:

CpaCyEcdsaVerify()
CpaCyEcdsaVerify()

typedef struct **_CpaCyEcdsaStats64 CpaCyEcdsaStats64**
typedef 结构_CpaCyEcdsaStats6CpaCyEcdsaStats6

Cryptographic ECDSA Statistics.
加密 ECDSA 统计。

This structure contains statistics on the Cryptographic ECDSA operations. Statistics are set to zero when the component is initialized, and are collected per instance.
此结构包含有关加密 ECDSA 操作的统计信息。当组件初始化时，统计信息被设置为零，并针对每个实例进行收集。

typedef void(* **CpaCyEcdsaGenSignCbFunc**)(void *pCallbackTag, **CpaStatus** status, void *pOpData
typedef void(* CpaCyEcdsaGenSignCbFufc)(void *pCallbackTag, **CpaStatus** status, void *pOpData

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

This is the prototype for the CpaCyEcdsaGenSignCbFunc callback function.

为许多 ECDSA Sign API 函数调用的通用回调函数的定义。这是 CpaCyEcdsaGenSignCbFunc 回调函

数的原型。

Context:
背景:

This callback function can be executed in a context that DOES NOT permit sleeping to occur.
这个回调函数可以在不允许休眠发生的上下文中执行。

Assumptions:
假设:

None
没有人

Side-Effects:

副作用:

None

没有人

Reentrant:

可重入:

No

不

Thread-safe:

线程安全:

Yes

是

Parameters:

参数:

[in] *pCallbackTag* User-supplied value to help identify request.

[in] pCallbackTag 使用者提供的值，可协助识别要求。

[in] *status* Status of the operation. Valid values are CPA_STATUS_SUCCESS, CPA_STATUS_FAIL and CPA_STATUS_UNSUPPORTED.

[in] status 操作的状态。有效值为 CPA_STATUS_SUCCESS、CPA_STATUS_FAIL 和 CPA_STATUS_UNSUPPORTED。

[in] *pOpData* Opaque pointer to Operation data supplied in request.

[in]指向请求中提供的操作数据的 pOpData Opaque 指针。

[in] *multiplyStatus* Status of the point multiplication.

[in]点乘法的多状态状态。

[in] *pOut* Output data from the request.

[in]从请求中输出数据。

Return values:

返回值:

None

没有人

Precondition:

前提条件:

Component has been initialized.

组件已初始化。

Postcondition:

后置条件:

None

没有人

Note:

注意:

None

没有人

See also:

另请参见:

cpaCyEcdsaSignR() cpaCyEcdsaSignS()

cpaCyEcdsaSignR() cpaCyEcdsaSignS()

typedef void(* **CpaCyEcdsaSignRSCbFunc**)(void *pCallbackTag, **CpaStatus** status, void *pOpData,

typedef void(* **CpaCyEcdsaSignRSCbFunc**)(void *pCallbackTag, **CpaStatus** status, void *pOpData,

Definition of callback function invoked for cpaCyEcdsaSignRS requests.

为 cpaCyEcdsaSignRS 请求调用的回调函数的定义。

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

这是 CpaCyEcdsaSignRSCbFunc 回调函数的原型，它将提供 ECDSA 消息签名 r 和 s 参数。

Context:

背景:

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

这个回调函数可以在不允许休眠发生的上下文中执行。

Assumptions:

假设:

None

没有人

Side-Effects:

副作用:

None

没有人

Reentrant:

可重入:

No

不

Thread-safe:

线程安全:

Yes

是

Parameters:

参数:

[in] *pCallbackTag* User-supplied value to help identify request.

[in] pCallbackTag 使用者提供的值，可协助识别要求。

[in] *status* Status of the operation. Valid values are CPA_STATUS_SUCCESS, CPA_STATUS_FAIL and CPA_STATUS_UNSUPPORTED.

[in] status 操作的状态。有效值为 CPA_STATUS_SUCCESS、CPA_STATUS_FAIL 和 CPA_STATUS_UNSUPPORTED。

[in] *pOpData* Operation data pointer supplied in request.

[in]请求中提供了 pOpData 操作数据指针。

[in] *multiplyStatus* Status of the point multiplication.

[in]点乘法的多状态状态。

[in] *pR* Ecdsa message signature r.

pR Ecdsa 消息签名。

[in] *pS* Ecdsa message signature s.

[in] pS Ecdsa 消息签名

Return values:
返回值:

None
没有人

Precondition:
前提条件:

Component has been initialized.
组件已初始化。

Postcondition:
后置条件:

None
没有人

Note:

注意:

None
没有人

See also:

另请参见:

cpaCyEcdsaSignRS()
cpaCyEcdsaSignRS()

typedef void(* **CpaCyEcdsaVerifyCbFunc**)(void *pCallbackTag, **CpaStatus** status, void *pOpData,

typedef void(* **CpaCyEcdsaVerifyCbFunc**)(void *pCallbackTag, **CpaStatus** status, void *pOpData,

Definition of callback function invoked for cpaCyEcdsaVerify requests.

This is the prototype for the CpaCyEcdsaVerifyCbFunc callback function.

为 cpaCyEcdsaVerify 请求调用的回调函数的定义。这是

CpaCyEcdsaVerifyCbFunc 回调函数的原型。

Context:
背景:

This callback function can be executed in a context that DOES NOT permit sleeping to occur.
这个回调函数可以在不允许休眠发生的上下文中执行。

Assumptions:
假设:

None
没有人

Side-Effects:
副作用:

None
没有人

Reentrant:
可重入:

No
不

Thread-safe:
线程安全:

Yes
是

Parameters:
参数:

`[in]` *pCallbackTag* User-supplied value to help identify request.
[in] pCallbackTag 使用者提供的值，可协助识别要求。

`[in]` *status* Status of the operation. Valid values are CPA_STATUS_SUCCESS, CPA_STATUS_FAIL and CPA_STATUS_UNSUPPORTED.
[in] status 操作的状态。有效值为 CPA_STATUS_SUCCESS、CPA_STATUS_FAIL 和 CPA_STATUS_UNSUPPORTED。

`[in]` *pOpData* Operation data pointer supplied in request.
[in]请求中提供了 pOpData 操作数据指针。

`[in]` *verifyStatus* The verification status.
[in] verifyStatus 验证状态。

Return values:
返回值:

None
没有人

Precondition:
前提条件:

Component has been initialized.

组件已初始化。

Postcondition:
后置条件:

None
没有人

Note:

注意:

None
没有人

See also:

另请参见:

cpaCyEcdsaVerify()
cpaCyEcdsaVerify()

16.10 Function Documentation

16.11 功能文档

```
                 cpaCyEcdsaSignR  (const                            instanceHandle,
                 cpaCyEcdsaSignR  (const                            instanceHandle,
const                                                               pCb,

                              void *pCallbackTag,
                              const  pOpData,
                              *
                              *                                     pSignStatus, pR
                              *
                           )
```

Generate ECDSA Signature R.
生成 ECDSA 签名 r。

This function generates ECDSA Signature R as per ANSI X9.62 2005 section 7.3.
该函数根据 ANSI X9.62 2005 第 7.3 节生成 ECDSA 签名 R。

Context:
背景:

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.
当作为异步函数调用时，它不能休眠。它可以在不允许休眠的上下文中执行。当作为同步函数调用时，它可能会休眠。它不能在不允许休眠的上下文中执行。

Assumptions:
假设:

None
没有人

Side-Effects:
副作用:

None
没有人

Blocking:
阻止:

Yes when configured to operate in synchronous mode.
当配置为在同步模式下运行时，是。

Reentrant:
可重入:

No
不

Thread-safe:
线程安全:

Yes
是

Parameters:
参数:

`[in]` *instanceHandle* Instance handle.
[in] instanceHandle 执行个体控制代码。

`[in]` *pCb* Callback function pointer. If this is set to a NULL value the function will operate synchronously.
[in] pCb 回调函数指针。如果设置为空值，函数将同步运行。

`[in]` *pCallbackTag* User-supplied value to help identify request.
[in] pCallbackTag 使用者提供的值，可协助识别要求。

`[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.
[in] pOpData 结构，包含执行作业所需的所有资料。客户端代码为此结构分配内存。该组件取得内存的所有权，直到它在回调中被返回。

`[out]` *pSignStatus* In synchronous mode, the multiply output is valid (CPA_TRUE) or the output is invalid (CPA_FALSE).

[out] pSignStatus 在同步模式下，乘法输出有效(CPA_TRUE)或输出无效(CPA_FALSE)。

`[out]` *pR* ECDSA message signature r.

[out] pR ECDSA 消息签名 r。

Return values:
返回值:

CPA_STATUS_SUCCESS Function executed successfully.

CPA_STATUS_SUCCESS 函数执行成功。

CPA_STATUS_FAIL Function failed.

CPA_STATUS_FAIL 函数失败。

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.

CPA_STATUS_RETRY 重新提交请求。传递的 CPA_STATUS_INVALID_PARAM 参数无效。与系统资源相关的 CPA_STATUS_RESOURCE 错误。CPA_STATUS_RESTARTING API 实现正在重新启动。重新提交请求。不支持 CPA_STATUS_UNSUPPORTED 函数。

Precondition:

前提条件:

The component has been initialized via cpaCyStartInstance function.

该组件已通过 cpaCyStartInstance 函数初始化。

Postcondition:
后置条件:

None

没有人

Note:

注意:

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.
当 pCb 为非空时，会生成一个异步回调来响应此函数调用。为了获得最佳性能，数据指针应该 8 字节对齐。

See also:

另请参见:

None
没有人

```
                  cpaCyEcdsaSignS  (const                          instanceHandle
                  cpaCyEcdsaSignS  (const                          instanceHandle,
const                                                              pCb,

                               void *pCallbackTag,
                               const pOpData,
                               *
                               *                                   pSignStatus, pS
                               *
                            )
```

Generate ECDSA Signature S.
生成 ECDSA 签名。

This function generates ECDSA Signature S as per ANSI X9.62 2005 section 7.3.
该函数根据 ANSI X9.62 2005 第 7.3 节生成 ECDSA 签名。

Context:
背景:

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.
当作为异步函数调用时，它不能休眠。它可以在不允许休眠的上下文中执行。当作为同步函数调用时，它可能会休眠。它不能在不允许休眠的上下文中执行。

Assumptions:
假设:

None
没有人

Side-Effects:
副作用:

None
没有人

Blocking:
阻止:

Yes when configured to operate in synchronous mode.
当配置为在同步模式下运行时，是。

Reentrant:
可重入：

No
不

Thread-safe:

线程安全:

Yes
是

Parameters:
参数:

[in] *instanceHandle* Instance handle.
[in] instanceHandle 执行个体控制代码。
[in] *pCb* Callback function pointer. If this is set to a NULL value the function will operate synchronously.
[in] pCb 回调函数指针。如果设置为空值，函数将同步运行。
[in] *pCallbackTag* User-supplied value to help identify request.
[in] pCallbackTag 使用者提供的值，可协助识别要求。
[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.
[in] pOpData 结构，包含执行作业所需的所有资料。客户端代码为此结构分配内存。该组件取得内存的所有权，直到它在回调中被返回。
[out] *pSignStatus* In synchronous mode, the multiply output is valid (CPA_TRUE) or the output is invalid (CPA_FALSE).
[out] pSignStatus 在同步模式下，乘法输出有效(CPA_TRUE)或输出无效(CPA_FALSE)。
[out] *pS* ECDSA message signature s.
[out] pS ECDSA 消息签名。

Return values:
返回值:

CPA_STATUS_SUCCESS Function executed successfully.
CPA_STATUS_SUCCESS 函数执行成功。
CPA_STATUS_FAIL Function failed.
CPA_STATUS_FAIL 函数失败。
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.

CPA_STATUS_RETRY 重新提交请求。传递的 CPA_STATUS_INVALID_PARAM 参数无效。与系统资源相关的 CPA_STATUS_RESOURCE 错误。CPA_STATUS_RESTARTING API 实现正在重新启动。重新提交请求。不支持 CPA_STATUS_UNSUPPORTED 函数。

Precondition:

前提条件:

The component has been initialized via cpaCyStartInstance function.
该组件已通过 cpaCyStartInstance 函数初始化。

Postcondition:
后置条件:

None
没有人

Note:

注意：

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.
当pCb为非空时，会生成一个异步回调来响应此函数调用。为了获得最佳性能，数据指针应该8字节对齐。

See also:

另请参见:

None
没有人

cpaCyEcdsaSignRS (const instanceHandle

```
cpaCyEcdsaSignRS (  const                                           instanceHandle,
                          const pCb,
                          void *pCallbackTag,
                          const * pOpData,
                          *pSignStatus,
                          *pR,
                          *pS
                         )
```

Generate ECDSA Signature R & S.
生成ECDSA签名R & S。

This function generates ECDSA Signature R & S as per ANSI X9.62 2005 section 7.3.
此函数根据ANSI X9.62 2005第7.3节生成ECDSA签名R & S。

Context:
背景:

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.
当作为异步函数调用时，它不能休眠。它可以在不允许休眠的上下文中执行。当作为同步函数调用时，它可能会休眠。它不能在不允许休眠的上下文中执行。

Assumptions:

假设:

None

没有人

Side-Effects:

副作用:

None

没有人

Blocking:

阻止:

Yes when configured to operate in synchronous mode.

当配置为在同步模式下运行时，是。

Reentrant:

可重入:

No

不

Thread-safe:

线程安全:

Yes

是

Parameters:

参数:

`[in]` *instanceHandle* Instance handle.

[in] instanceHandle 执行个体控制代码。

`[in]` *pCb* Callback function pointer. If this is set to a NULL value the function will operate synchronously.

[in] pCb 回调函数指针。如果设置为空值，函数将同步运行。

`[in]` *pCallbackTag* User-supplied value to help identify request.

[in] pCallbackTag 使用者提供的值，可协助识别要求。

`[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.

[in] pOpData 结构，包含执行作业所需的所有资料。客户端代码为此结构分配内存。该组件取得内存的所有权，直到它在回调中被返回。

`[out]` *pSignStatus* In synchronous mode, the multiply output is valid (CPA_TRUE) or the output is invalid (CPA_FALSE).

[out] pSignStatus 在同步模式下，乘法输出有效(CPA_TRUE)或输出无效(CPA_FALSE)。

`[out]` *pR* ECDSA message signature r.

[out] pR ECDSA 消息签名 r。

`[out]` *pS* ECDSA message signature s.

[out] pS ECDSA 消息签名。

Return values:

返回值:

CPA_STATUS_SUCCESS Function executed successfully.

CPA_STATUS_SUCCESS 函数执行成功。

CPA_STATUS_FAIL Function failed.

CPA_STATUS_FAIL 函数失败。

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.

CPA_STATUS_RETRY 重新提交请求。传递的 CPA_STATUS_INVALID_PARAM 参数无效。与系统资源相关的 CPA_STATUS_RESOURCE 错误。CPA_STATUS_RESTARTING API 实现正在重新启动。重新提交请求。不支持 CPA_STATUS_UNSUPPORTED 函数。

Precondition:

前提条件:

The component has been initialized via cpaCyStartInstance function.

该组件已通过 cpaCyStartInstance 函数初始化。

Postcondition:

后置条件:

None

没有人

Note:

注意:

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.

当 pCb 为非空时，会生成一个异步回调来响应此函数调用。为了获得最佳性能，数据指针应该 8 字节对齐。

See also:

另请参见:

None

没有人

```
cpaCyEcdsaVerify (  const                                       instanceHandle,
                          const pCb,
                          void *pCallbackTag,
                          const *  pOpData,
                          *pVerifyStatus
                         )
```

```
cpaCyEcdsaVerify (  const                                       instanceHandle,
                          const pCb,
                          void *pCallbackTag,
                          const * pOpData,
                          *pVerifyStatus
                         )
```

Verify ECDSA Public Key.
验证ECDSA公钥。

This function performs ECDSA Verify as per ANSI X9.62 2005 section 7.4.
该函数根据ANSI X9.62 2005第7.4节执行ECDSA验证。

A response status of ok (verifyStatus == CPA_TRUE) means that the signature was verified
响应状态为ok (verifyStatus == CPA_TRUE)意味着签名已经过验证

Context:
背景:

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.
当作为异步函数调用时，它不能休眠。它可以在不允许休眠的上下文中执行。当作为同步函数调用时，它可能会休眠。它不能在不允许休眠的上下文中执行。

Assumptions:
假设:

None
没有人

Side-Effects:
副作用:

None
没有人

Blocking:
阻止:

Yes when configured to operate in synchronous mode.
当配置为在同步模式下运行时，是。

Reentrant:
可重入:

No
不

Thread-safe:

线程安全:

Yes

是

Parameters:

参数:

[in] *instanceHandle* Instance handle.

[in] instanceHandle 执行个体控制代码。

[in] *pCb* Callback function pointer. If this is set to a NULL value the function will operate synchronously.

[in] pCb 回调函数指针。如果设置为空值，函数将同步运行。

[in] *pCallbackTag* User-supplied value to help identify request.

[in] pCallbackTag 使用者提供的值，可协助识别要求。

[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.

[in] pOpData 结构，包含执行作业所需的所有资料。客户端代码为此结构分配内存。该组件取得内存的所有权，直到它在回调中被返回。

[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.

[out] pVerifyStatus 在同步模式下，如果点不在曲线上或不在无穷远处，则设置为 CPA_FALSE。如果点在曲线上，则设置为 CPA_TRUE。

Return values:

返回值:

CPA_STATUS_SUCCESS Function executed successfully.

CPA_STATUS_SUCCESS 函数执行成功。

CPA_STATUS_FAIL Function failed.

CPA_STATUS_FAIL 函数失败。

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.

CPA_STATUS_RETRY 重新提交请求。传递的 CPA_STATUS_INVALID_PARAM 参数无效。与系统资源相关的 CPA_STATUS_RESOURCE 错误。CPA_STATUS_RESTARTING API 实现正在重新启动。重新提交请求。不支持 CPA_STATUS_UNSUPPORTED 函数。

Precondition:

前提条件:

The component has been initialized via cpaCyStartInstance function.

该组件已通过 cpaCyStartInstance 函数初始化。

Postcondition:

后置条件:

None

没有人

Note:

注意:

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.
当 pCb 为非空时，会生成一个 CpaCyEcdsaVerifyCbFunc 类型的异步回调来响应此函数调用。为了获得最佳性能，数据指针应该 8 字节对齐。

See also:

另请参见:

CpaCyEcdsaVerifyOpData, **CpaCyEcdsaVerifyCbFunc**
CpaCyEcdsaVerifyOpData, CpaCyEcdsaVerifyCbFunc

CpaStatus cpaCyEcdsaQueryStats64 (const **CpaInstanceHandle** *instanceHandle*, **CpaCyEcdsaStats64** * *pEcdsaStats*

CpaStatus cpaCyEcdsaQueryStats64 (const **CpaInstanceHandle** *instanceHandle*, CpaCyEcdsaStats64 * *pEcdsaStats*

Query statistics for a specific ECDSA instance.
查询特定 ECDSA 实例的统计信息。

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.
该函数将查询 ECDSA 实现的特定实例的统计信息。用户必须分配 CpaCyEcdsaStats64 结构，并将对该结构的引用传递到此函数调用中。此函数将统计结果写入传入的 CpaCyEcdsaStats64 结构中。

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.
注意:此函数返回的统计数据不会中断当前的数据处理，因此可能会与统计数据检索过程中正在进行的操作稍微不同步。

Context:
背景:

This is a synchronous function and it can sleep. It MUST NOT be executed in a context that DOES NOT permit sleeping.
这是一个同步功能，它可以休眠。它不能在不允许休眠的上下文中执行。

Assumptions:
假设:

None
没有人

Side-Effects:
副作用:

None
没有人

Blocking:
阻止:

This function is synchronous and blocking.
这个函数是同步的和阻塞的。

Reentrant:
可重入:

No

不

Thread-safe:

线程安全:

Yes

是

Parameters:

参数:

`[in]`	*instanceHandle*	Instance handle.
[in] instanceHandle 执行个体控制代码。		
`[out]`	*pEcdsaStats*	Pointer to memory into which the statistics will be written.
[out]pecd stats 指向将写入统计信息的内存的指针。		

Return values:

返回值:

CPA_STATUS_SUCCESS	Function executed successfully.
CPA_STATUS_SUCCESS 函数执行成功。	
CPA_STATUS_FAIL	Function failed.
CPA_STATUS_INVALID_PARAM	Invalid parameter passed in.
CPA_STATUS_RESOURCE	Error related to system resources.

CPA_STATUS_FAIL 函数失败。传递的 CPA_STATUS_INVALID_PARAM 参数无效。与系统资源相关的 CPA_STATUS_RESOURCE 错误。

CPA_STATUS_RESTARTING	API implementation is restarting. Resubmit the request.
CPA_STATUS_RESTARTING API 实现正在重新启动。重新提交请求。	
CPA_STATUS_UNSUPPORTED	Function is not supported.
不支持 CPA_STATUS_UNSUPPORTED 函数。	

Precondition:

前提条件:

Component has been initialized.

组件已初始化。

Postcondition:
后置条件:

None
没有人

Note:

注意:

This function operates in a synchronous manner and no asynchronous callback will be generated.
该函数以同步方式运行，不会生成异步回调。

See also:

另请参见:

CpaCyEcdsaStats64
CpaCyEcdsaStats64

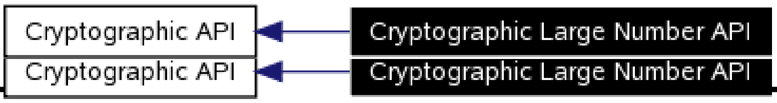
20 Cryptographic Large Number API

21 加密大数API

[Cryptographic API]

[Cryptographic API]

Collaboration diagram for Cryptographic Large Number API:
加密大数API的协作图:


Cryptographic API
Cryptographic Large Number API
Cryptographic API
Cryptographic Large Number API

21.1 Detailed Description

21.2 详细描述

File: cpa_cy_ln.h

文件:cpa_cy_ln.h

These functions specify the Cryptographic API for Large Number Operations.
这些函数为大量操作指定加密API。

Note:
注意:

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.
在QuickAssist API上，大数用八位字节字符串表示，存储在**CpaFlatBuffer**

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.
一个n位数字n的取值范围是从2^(n-1到2^(n)-1.换句话说，它的最高有效位n-1位(从零开始计数)必须设置为1。我们还可以说，数N的位长N由n = floor(log2(N))+1定义。

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).
存储N位数N的缓冲器b必须“足够大”。换句话说，b.dataLenInBytes至少必须是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.
该数字以“大端”格式存储。这意味着最低有效字节(LSB)是b[b.dataLenInBytes-1]，而最高有效字节(MSB)是b[b . dataleninbytes-minLenInBytes]。在缓冲区大小“完全”正确的情况下，MSB为b[0]。否则，从b[0]到MSB的所有字节都必须设为0x00。

The largest bit-length we support today is 4096 bits. In other words, we can deal with numbers up to a value of (2^4096)-1.
我们今天支持的最大位长是4096位。换句话说，我们可以处理数值高达(2^4096)-1.)的数字

21.3 Data Structures

21.4 数据结构

- struct **_CpaCyLnModExpOpData**

- 结构体_CpaCyLnModExpOpData
- struct **_CpaCyLnModInvOpData**
- 结构体_CpaCyLnModInvOpData
- struct **_CpaCyLnStats**
- 结构体_CpaCyLnStats
- struct **_CpaCyLnStats64**
- 结构体_CpaCyLnStats64

21.5 Typedefs

21.6 类型定义

- typedef **_CpaCyLnModExpOpData CpaCyLnModExpOpData**

- 数据类型说明_CpaCyLnModExpOpData CpaCyLnModExpOpData
- typedef **_CpaCyLnModInvOpData CpaCyLnModInvOpData**
- 数据类型说明_CpaCyLnModInvOpData CpaCyLnModInvOpData
- typedef **_CpaCyLnStats CPA_DEPRECATED**
- 数据类型说明_CpaCyLnStats CPA_DEPRECATED
- typedef **_CpaCyLnStats64 CpaCyLnStats64**
- 数据类型说明_CpaCyLnStats64 CpaCyLnStats64

21.7 Functions

21.8 功能

17.4 Functions

17.5 功能

- **CpaStatus cpaCyLnModExp** (const **CpaInstanceHandle** instanceHandle, const

- CpaStatus cpaCyLnModExp（常量 CpaInstanceHandle
  CpaCyGenFlatBufCbFunc pLnModExpCb, void *pCallbackTag, const **CpaCyLnModExpOpData**
  CpaCyGenFlatBufCbFunc pLnModExpCb，void *pCallbackTag，constCpaCyLnModExpOpData
  *pLnModExpOpData, **CpaFlatBuffer** *pResult)
  * pLnModExpOpData，CpaFlatBuffer
- **CpaStatus cpaCyLnModInv** (const **CpaInstanceHandle** instanceHandle, const
- CpaStatus cpaCyLnModInv（常量 CpaInstanceHandle
  CpaCyGenFlatBufCbFunc pLnModInvCb, void *pCallbackTag, const **CpaCyLnModInvOpData**
  CpaCyGenFlatBufCbFunc pLnModInvCb，void *pCallbackTag，constCpaCyLnModInvOpData
  *pLnModInvOpData, **CpaFlatBuffer** *pResult)
  *pLnModInvOpData，CpaFlatBuffer
- **CpaStatus CPA_DEPRECATED cpaCyLnStatsQuery** (const **CpaInstanceHandle** instanceHandle, struct **_CpaCyLnStats** *pLnStats)
- CpaStatus CPA_DEPRECATED cpaCyLnStatsQuery（常量 CpaInstanceHandle _CpaCyLnStats
- **CpaStatus cpaCyLnStatsQuery64** (const **CpaInstanceHandle** instanceHandle, **CpaCyLnStats64**
- CpaStatus cpaCyLnStatsQuery64（常量 CpaInstanceHandle CpaCyLnStats64
  *pLnStats)
  *计划状态)

17.6 Data Structure Documentation

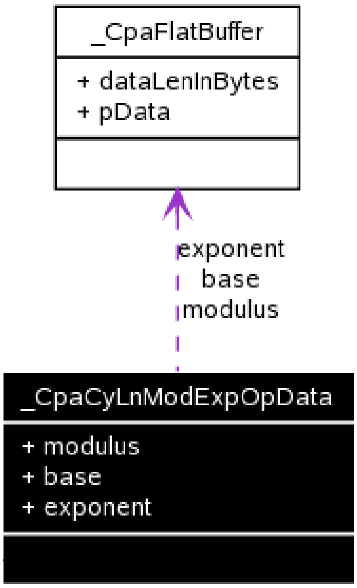
17.7 数据结构文档

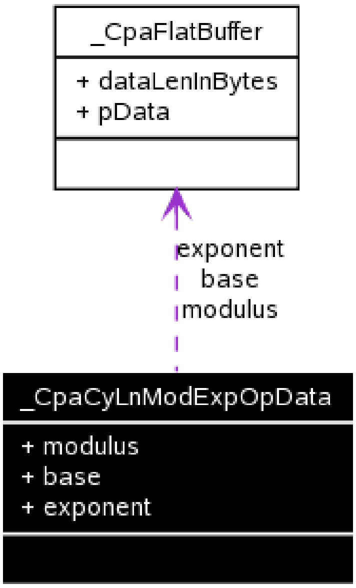
17.7.1 _CpaCyLnModExpOpData Struct Reference

17.7.2 _ CpaCyLnModExpOpData 结构引用

Collaboration diagram for _CpaCyLnModExpOpData:

_ CpaCyLnModExpOpData 的协作图:





17.7.2.1 Detailed Description
17.7.2.2 详细描述

Modular Exponentiation Function Operation Data.
模幂函数运算数据。

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.
此结构列出了cpaCyLnModExp函数中所需的不同项目。客户端必须为这个结构分配内存。当结构被传递给函数时，内存的所有权就传递给了函数。当这个结构在回调中返回时，内存的所有权返回给客户端。以位为单位的运算大小等于以下两者中最大的一个:模数、底数或指数。

Note:
注意:

> 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.
> 如果客户端在将此结构中引用的内存提交给cpaCyLnModExp函数之后，在回调中返回之前修改或释放该内存，将导致未定义的行为。

17.5.1 _CpaCyLnModExpOpData Struct Reference

17.5.2_ CpaCyLnModExpOpData 结构引用

The values of the base, the exponent and the modulus MUST all be less than 2^4096, and the modulus must not be equal to zero.

底数、指数和模数的值都必须小于 2ˆ4096，并且模数不能等于零。

17.7.2.3 Data Fields
17.7.2.4 数据字段

- **CpaFlatBuffer modulus**
- CpaFlatBuffer modulus
- **CpaFlatBuffer base**
- CpaFlatBuffer base
- **CpaFlatBuffer exponent**
- CpaFlatBuffer exponent

17.7.2.5 Field Documentation
17.7.2.6 现场文件

CpaFlatBuffer _CpaCyLnModExpOpData::modulus

Flat buffer containing a pointer to the modulus. This number may be up to 4096 bits in length, and MUST be greater than zero.
包含模数指针的平面缓冲区。这个数字最长可达 4096 位，并且必须大于零。

CpaFlatBuffer _CpaCyLnModExpOpData::base

Flat buffer containing a pointer to the base. This number may be up to 4096 bits in length.
包含指向基底的指针的平面缓冲区。这个数字的长度可以达到 4096 位。

CpaFlatBuffer _CpaCyLnModExpOpData::exponent

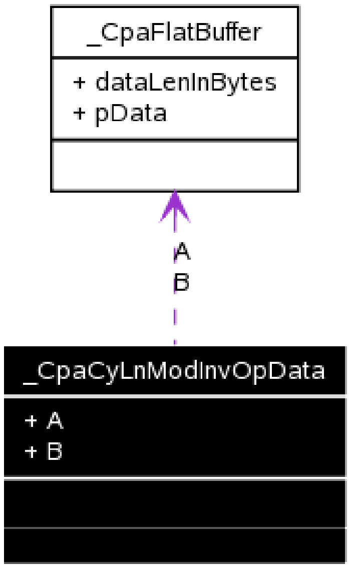
Flat buffer containing a pointer to the exponent. This number may be up to 4096 bits in length.
包含指数指针的平面缓冲区。这个数字的长度可以达到 4096 位。

17.5.3 _CpaCyLnModInvOpData Struct Reference

17.5.4 _CpaCyLnModInvOpData 结构引用

Collaboration diagram for _CpaCyLnModInvOpData:

_ CpaCyLnModInvOpData 的协作图:



17.5.4.1 Detailed Description
17.5.4.2 详细描述

Modular Inversion Function Operation Data.
模逆函数运算数据。

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.
此结构列出了函数中所需的不同项目 cpaCyLnModInv

Note:
注意:

17.5.2 _CpaCyLnModInvOpData Struct Reference

17.5.3_CpaCyLnModInvOpData 结构引用

> 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.
>
> 如果客户端在将此结构中引用的内存提交给 cpaCyLnModInv 函数之后，在回调中返回之前修改或释放该内存，将导致未定义的行为。

Note that the values of A and B MUST NOT both be even numbers, and both MUST be less than 2^4096.
请注意，a 和 b 的值不能都是偶数，并且都必须小于 2^4096.

17.5.4.3 Data Fields
17.5.4.4 数据字段

- **CpaFlatBuffer A**
- CpaFlatBuffer A
- **CpaFlatBuffer B**
- CpaFlatBuffer B

17.5.4.5 Field Documentation
17.5.4.6 现场文件

CpaFlatBuffer _CpaCyLnModInvOpData::A

CpaFlatBuffer _CpaCyLnModInvOpData::A

Flat buffer containing a pointer to the value that will be inverted. This number may be up to 4096 bits in length, it MUST NOT be zero, and it MUST be co-prime with B.
包含将被反转的值的指针的平面缓冲区。这个数的长度可以达到 4096 位，它不能为零，并且必须与 b 互质。

CpaFlatBuffer _CpaCyLnModInvOpData::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 4096 bits in length, it MUST NOT be zero, and it MUST be co-prime with A.
包含将用作模数的值的指针的平面缓冲区。这个数的长度可以达到 4096 位，它不能为零，并且必须与 a 互质。

17.5.4 _CpaCyLnStats Struct Reference
17.5.5 _CpaCyLnStats 结构引用

17.5.5.1 Detailed Description
17.5.5.2 详细描述

Look Aside Cryptographic large number Statistics.
撇开加密大数统计不谈。

Deprecated:
Deprecated:

As of v1.3 of the Crypto API, this structure has been deprecated, replaced by **CpaCyLnStats64**.
从Crypto API的1.3版开始，这种结构已被取代，由CpaCyLnStats64

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.
此结构包含有关旁视加密大数操作的统计信息。当组件初始化时，统计信息被设置为零，并针对每个实例进行收集。

17.5.5.3 Data Fields
17.5.5.4 数据字段

- **Cpa32U numLnModExpRequests**
- Cpa32U numLnModExpRequests
- **Cpa32U numLnModExpRequestErrors**
- Cpa32U numLnModExpRequestErrors
- **Cpa32U numLnModExpCompleted**
- Cpa32U numLnModExpCompleted
- **Cpa32U numLnModExpCompletedErrors**
- Cpa32U numLnModExpCompletedErrors
- **Cpa32U numLnModInvRequests**
- Cpa32U numLnModInvRequests
- **Cpa32U numLnModInvRequestErrors**
- Cpa32U numLnModInvRequestErrors
- **Cpa32U numLnModInvCompleted**
- Cpa32U numLnModInvCompleted
- **Cpa32U numLnModInvCompletedErrors**
- Cpa32U numLnModInvCompletedErrors

17.5.5.5 Field Documentation
17.5.5.6 现场文件

Cpa32U _CpaCyLnStats::numLnModExpRequests

Total number of successful large number modular exponentiation requests.
成功的大数模幂运算请求的总数。

Cpa32U _CpaCyLnStats::numLnModExpRequestErrors

Total number of large number modular exponentiation requests that had an error and could not be
有错误且无法处理的大数模幂运算请求的总数

processed.

17.5.4_CpaCyLnStats 结构引用已处理。

Total number of large number modular exponentiation operations that completed successfully.
成功完成的大数模幂运算的总数。

Total number of large number modular exponentiation operations that could not be completed successfully due to errors.
由于错误而无法成功完成的大数模幂运算的总数。

Total number of successful large number modular inversion requests.
成功的大数模块化反转请求的总数。

Total number of large number modular inversion requests that had an error and could not be processed.
出现错误且无法处理的大数模块化反转请求的总数。

Total number of large number modular inversion operations that completed successfully.
成功完成的大数取模求逆操作的总数。

Total number of large number modular inversion operations that could not be completed successfully due to errors.
由于错误而无法成功完成的大数取模求逆操作的总数。

17.5.5 _CpaCyLnStats64 Struct Reference

17.5.6 _ CpaCyLnStats64 结构引用

17.5.6.1 Detailed Description

17.5.6.2 详细描述

Look Aside Cryptographic large number Statistics.
撇开加密大数统计不谈。

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.
此结构包含有关旁视加密大数操作的统计信息。当组件初始化时，统计信息被设置为零，并针对每个实例进行收集。

17.5.6.3 Data Fields

17.5.6.4 数据字段

- **Cpa64U numLnModExpRequests**
- Cpa64U numLnModExpRequests
- **Cpa64U numLnModExpRequestErrors**

- Cpa64U numLnModExpRequestErrors
- **Cpa64U numLnModExpCompleted**
- Cpa64U numLnModExpCompleted
- **Cpa64U numLnModExpCompletedErrors**
- Cpa64U numLnModExpCompletedErrors
- **Cpa64U numLnModInvRequests**
- Cpa64U numLnModInvRequests
- **Cpa64U numLnModInvRequestErrors**
- Cpa64U numLnModInvRequestErrors
- **Cpa64U numLnModInvCompleted**
- Cpa64U numLnModInvCompleted
- **Cpa64U numLnModInvCompletedErrors**
- Cpa64U numLnModInvCompletedErrors

17.5.6.5 Field Documentation

17.5.6.6 现场文件

Cpa64U CpaCyLnStats64::numLnModExpRequests

Cpa64U CpaCyLnStats64::numLnModExpRequests

Total number of successful large number modular exponentiation requests.
成功的大数模幂运算请求的总数。

Cpa64U CpaCyLnStats64::numLnModExpRequestErrors

Cpa64U CpaCyLnStats64::numLnModExpRequestErrors

Total number of large number modular exponentiation requests that had an error and could not be processed.
出现错误且无法处理的大数模幂运算请求的总数。

Total number of large number modular exponentiation operations that completed successfully.
成功完成的大数模幂运算的总数。

Cpa64U _CpaCyLnStats64::numLnModExpCompletedErrors

Total number of large number modular exponentiation operations that could not be completed successfully due to errors.
由于错误而无法成功完成的大数模幂运算的总数。

Cpa64U _CpaCyLnStats64::numLnModInvRequests

Total number of successful large number modular inversion requests.
成功的大数模块化反转请求的总数。

Cpa64U _CpaCyLnStats64::numLnModInvRequestErrors

Total number of large number modular inversion requests that had an error and could not be processed.
出现错误且无法处理的大数模块化反转请求的总数。

Cpa64U _CpaCyLnStats64::numLnModInvCompleted

Total number of large number modular inversion operations that completed successfully.
成功完成的大数取模求逆操作的总数。

Cpa64U _CpaCyLnStats64::numLnModInvCompletedErrors

Total number of large number modular inversion operations that could not be completed successfully due to errors.
由于错误而无法成功完成的大数取模求逆操作的总数。

17.8 Typedef Documentation

17.9 Typedef 文档

typedef struct **_CpaCyLnModExpOpData CpaCyLnModExpOpData**

Modular Exponentiation Function Operation Data.
模幂函数运算数据。

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.
此结构列出了cpaCyLnModExp函数中所需的不同项目。客户端必须为这个结构分配内存。当结构被传递给函数时，内存的所有权就传递给了函数。当这个结构在回调中返回时，内存的所有权返回给客户端。以位为单位的运算大小等于以下两者中最大的一个:模数、底数或指数。

Note:
注意:

> 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.
> 如果客户端在将此结构中引用的内存提交给cpaCyLnModExp函数之后，在回调中返回之前修改或释放该内存，将导致未定义的行为。

The values of the base, the exponent and the modulus MUST all be less than 2^4096, and the modulus must not be equal to zero.
底数、指数和模数的值都必须小于 2^4096，并且模数不能等于零。

typedef struct **_CpaCyLnModInvOpData CpaCyLnModInvOpData**

typedef struct _CpaCyLnModInvOpData CpaCyLnModInvOpData

Modular Inversion Function Operation Data.
模逆函数运算数据。

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.
此结构列出了函数中所需的不同项目 cpaCyLnModInv

Note:
注意:

> 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.
> 如果客户端在将此结构中引用的内存提交给 cpaCyLnModInv 函数之后，在回调中返回之前修改或释放该内存，将导致未定义的行为。

Note that the values of A and B MUST NOT both be even numbers, and both MUST be less than 2^4096.

请注意，a 和 b 的值不能都是偶数，并且都必须小于 2^4096.

typedef struct _CpaCyLnStats CPA_DEPRECATED

Look Aside Cryptographic large number Statistics.
撇开加密大数统计不谈。

Deprecated:
Deprecated:

As of v1.3 of the Crypto API, this structure has been deprecated, replaced by **CpaCyLnStats64**.
从 Crypto API 的 1.3 版开始，这种结构已被取代，由 CpaCyLnStats64

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.
此结构包含有关旁视加密大数操作的统计信息。当组件初始化时，统计信息被设置为零，并针对每个实例进行收集。

typedef struct _CpaCyLnStats64 CpaCyLnStats64

Look Aside Cryptographic large number Statistics.
撇开加密大数统计不谈。

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.
此结构包含有关旁视加密大数操作的统计信息。当组件初始化时，统计信息被设置为零，并针对每个实例进行收集。

17.8 Function Documentation

17.9 功能文档

```
cpaCyLnModExp (  const                                   instanceHandle,
                      const  pLnModExpCb, void *pCallbackTag,
                      const *  pLnModExpOpData,
                      *pResult

                     )
```

Perform modular exponentiation operation.
执行模幂运算。

This function performs modular exponentiation. It computes the following result based on the inputs:

result = (base ^ exponent) mod modulus

该函数执行模幂运算。它根据输入计算以下结果:结果=(基本^指数)模模数

Context:

背景:

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.

当作为异步函数调用时，它不能休眠。它可以在不允许休眠的上下文中执行。当作为同步函数调用时，它可能会休眠。它不能在不允许休眠的上下文中执行。

Assumptions:
假设:

None
没有人

Side-Effects:
副作用:

None
没有人

Reentrant:
可重入:

No
不

Thread-safe:
线程安全:

Yes
是

Parameters:

[in]	*instanceHandle*	Instance handle.
[in]	*pLnModExpCb*	Pointer to callback function to be invoked when the operation is complete.
[in]	*pCallbackTag*	

参数:

[在]	*instanceHandle*	实例句柄。
[在]	*pLnModExpCb*	操作完成时要调用的回调函数的指针。
[在]	*pCallbackTag*	

Opaque User Data for this specific call. Will be returned unchanged in the callback.

此特定呼叫的不透明用户数据。将在回调中不变地返回。

[in] *pLnModExpOpData* Structure containing all the data needed to perform the LN modular

[in]plnmodepopdata 结构，包含执行 LN 模组化所需的所有资料

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.

取幂运算。客户端代码为此结构分配内存。该组件取得内存的所有权，直到它在回调中被返回。

[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.

[out]指向平面缓冲区的 pResult 指针，该缓冲区包含由客户端分配的、结果将写入其中的内存的指针。所需存储器的大小必须大于或等于存储模数所需的大小。在调用时，回调函数将在 pOut 参数中包含这个参数。

Return values:

返回值:

CPA_STATUS_SUCCESS Function executed successfully.

CPA_STATUS_SUCCESS 函数执行成功。

CPA_STATUS_FAIL Function failed.

CPA_STATUS_FAIL 函数失败。

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.

CPA_STATUS_RETRY 重新提交请求。传递的 CPA_STATUS_INVALID_PARAM 参数无效。与系统资源相关的 CPA_STATUS_RESOURCE 错误。CPA_STATUS_RESTARTING API 实现正在重新启动。重新提交请求。不支持 CPA_STATUS_UNSUPPORTED 函数。

Precondition:

前提条件:

The component has been initialized.

组件已初始化。

Postcondition:

后置条件:

None

没有人

Note:

注意:

Whe
n
pLn
Mod

ExpCb 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.
当 pLnModExpCb 为非 null 时，将生成一个 CpaCyLnModExpCbFunc 类型的异步回调来响应此函数调用。处理过程中生成的任何错误都在回调中返回的结构中报告。

See also:

另请参见:

CpaCyLnModExpOpData, **CpaCyGenFlatBufCbFunc**

CpaCyLnModExpOpData, CpaCyGenFlatBufCbFunc

```
cpaCyLnModInv (  const                                          instanceHandle
cpaCyLnModInv  (  const                                         instanceHandle,
                           const   pLnModInvCb, void *pCallbackTag,
                           const *   pLnModInvOpData,
                           *pResult

                          )
```

Perform modular inversion operation.
执行模逆运算。

This function performs modular inversion. It computes the following result based on the inputs:

result = (1/A) mod B.

这个函数执行模逆运算。它根据输入计算以下结果:result = (1/A) mod B。

Context:
背景:

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.
当作为异步函数调用时，它不能休眠。它可以在不允许休眠的上下文中执行。当作为同步函数调用时，它可能会休眠。它不能在不允许休眠的上下文中执行。

Assumptions:
假设:

None
没有人

Side-Effects:

副作用:

None

没有人

Reentrant:

可重入:

No

不

Thread-safe:

线程安全:

Yes

是

Parameters:

参数:

`[in]` *instanceHandle* Instance handle.

[in] instanceHandle 执行个体控制代码。

`[in]` *pLnModInvCb* Pointer to callback function to be invoked when the operation is

[in]当作业为时，要叫用的回呼函式的 pLnModInvCb 指标

complete.

完成。

`[in]` *pCallbackTag* Opaque User Data for this specific call. Will be returned unchanged in the callback.

[in] pCallbackTag 此特定调用的不透明用户数据。将在回调中不变地返回。

`[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.

[in] pLnModInvOpData 结构，包含执行 LN 模组反转运算所需的所有资料。客户端代码为此结构分配内存。该组件取得内存的所有权，直到它在回调中被返回。

`[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.

[out]指向平面缓冲区的 pResult 指针，该缓冲区包含由客户端分配的、结果将写入其中的内存的指针。所需存储器的大小必须大于或等于存储模数所需的大小。在调用时，回调函数将在 pOut 参数中包含这个参数。

Return values:

返回值:

CPA_STATUS_SUCCESS Function executed successfully.

CPA_STATUS_SUCCESS 函数执行成功。

CPA_STATUS_FAIL Function failed.

CPA_STATUS_FAIL 函数失败。

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.

CPA_STATUS_RETRY 重新提交请求。传递的 CPA_STATUS_INVALID_PARAM 参数无效。与系统资源相关的 CPA_STATUS_RESOURCE 错误。CPA_STATUS_RESTARTING API 实现正在重新启动。重新提交请求。不支持 CPA_STATUS_UNSUPPORTED 函数。

Precondition:

前提条件:

The component has been initialized.
组件已初始化。

Postcondition:
后置条件:

None
没有人

Note:

注意:

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.

当 pLnModInvCb 为非 null 时，将生成一个 CpaCyLnModInvCbFunc 类型的异步回调来响应此函数调用。处理过程中生成的任何错误都在回调中返回的结构中报告。

See also:

另请参见:

CpaCyLnModInvOpData, **CpaCyGenFlatBufCbFunc**

CpaCyLnModInvOpData, CpaCyGenFlatBufCbFunc

CpaStatus CPA_DEPRECATED cpaCyLnStatsQuery (const **CpaInstanceHandle** *instanceHandle*, struct **_CpaCyLnStats** * *pLnStats*)

CpaStatus CPA_DEPRECATED cpaCyLnStatsQuery (const CpaInstanceHandle *instanceHandle*,
struct _CpaCyLnStats * *pLnStats*
)

Query statistics for large number operations

查询大量操作的统计数据

Deprecated:

Deprecated:

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

从 Crypto API 1.3 版开始，此函数已被弃用，由

cpaCyLnStatsQuery64().

cpaCyLnStatsQuery64()。

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.

这个函数将查询一个特定的实例句柄以获得大量的统计数据。用户必须分配 CpaCyLnStats 结构，并将对该结构的引用传递到此函数调用中。该函数将统计结果写入传入的 CpaCyLnStats 结构中。

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.

注意:此函数返回的统计数据不会中断当前的数据处理，因此可能会与统计数据检索过程中正在进行的操作稍微不同步。

Context:

背景:

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

这是一个同步功能，它可以休眠。它不能在不允许休眠的上下文中执行。

Assumptions:

假设:

None

没有人

Side-Effects:

副作用:

None

没有人

Reentrant:

可重入:

No

不

Thread-safe:

线程安全:

Yes

是

Parameters:

参数:

`[in]` *instanceHandle* Instance handle.

[in] instanceHandle 执行个体控制代码。

`[out]` *pLnStats* Pointer to memory into which the statistics will be written.

[out] pLnStats 指向将写入统计信息的内存的指针。

Return values:

返回值:

CPA_STATUS_SUCCESS Function executed successfully.

CPA_STATUS_SUCCESS 函数执行成功。

CPA_STATUS_FAIL Function failed.

CPA_STATUS_INVALID_PARAM Invalid parameter passed in.

CPA_STATUS_RESOURCE Error related to system resources.

CPA_STATUS_FAIL 函数失败。传递的 CPA_STATUS_INVALID_PARAM 参数无效。与系统资源相关的 CPA_STATUS_RESOURCE 错误。

CPA_STATUS_RESTARTING API implementation is restarting. Resubmit the request.

CPA_STATUS_RESTARTING API 实现正在重新启动。重新提交请求。

CPA_STATUS_UNSUPPORTED Function is not supported.

不支持 CPA_STATUS_UNSUPPORTED 函数。

Precondition:

前提条件:

Acceleration Services unit has been initialized.

加速服务单元已初始化。

Postcondition:

后置条件:

None

没有人

Note:

注意:

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

该函数以同步方式运行，不会生成异步回调。

See also:

另请参见:

CpaCyLnStats

CpaCyLnStats

CpaStatus cpaCyLnStatsQuery64 (const **CpaInstanceHandle** *instanceHandle*,

CpaStatus cpaCyLnStatsQuery64 (const **CpaInstanceHandle** *instanceHandle*,

)

Query statistics (64-bit version) for large number operations

)查询大量运算的统计信息(64位版本)

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.

该函数将查询特定的实例句柄，以获取64位版本的大数统计信息。用户必须分配CpaCyLnStats64结构，并将对该结构的引用传递到此函数调用中。该函数将统计结果写入传入的CpaCyLnStats64结构中。

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.
注意:此函数返回的统计数据不会中断当前的数据处理，因此可能会与统计数据检索过程中正在进行的操作稍微不同步。

Context:
背景:

This is a synchronous function and it can sleep. It MUST NOT be executed in a context that DOES NOT permit sleeping.
这是一个同步功能，它可以休眠。它不能在不允许休眠的上下文中执行。

Assumptions:
假设:

None
没有人

Side-Effects:
副作用:

None
没有人

Reentrant:
可重入:

No
不

Thread-safe:
线程安全:

Yes
是

Parameters:
参数:

[in] *instanceHandle* Instance handle.
[in] instanceHandle执行个体控制代码。
[out] *pLnStats* Pointer to memory into which the statistics will be written.
[out] pLnStats指向将写入统计信息的内存的指针。

Return values:

返回值:

CPA_STATUS_SUCCESS Function executed successfully.

CPA_STATUS_SUCCESS 函数执行成功。

CPA_STATUS_FAIL Function failed.

CPA_STATUS_INVALID_PARAM Invalid parameter passed in.

CPA_STATUS_RESOURCE Error related to system resources.

CPA_STATUS_FAIL 函数失败。传递的 CPA_STATUS_INVALID_PARAM 参数无效。与系统资源相关的 CPA_STATUS_RESOURCE 错误。

CPA_STATUS_RESTARTING API implementation is restarting. Resubmit the request.

CPA_STATUS_RESTARTING API 实现正在重新启动。重新提交请求。

CPA_STATUS_UNSUPPORTED Function is not supported.

不支持 CPA_STATUS_UNSUPPORTED 函数。

Precondition:

前提条件:

Acceleration Services unit has been initialized.

加速服务单元已初始化。

Postcondition:

后置条件:

None

没有人

Note:

注意:

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

该函数以同步方式运行，不会生成异步回调。

See also:

另请参见:

CpaCyLnStats

CpaCyLnStats

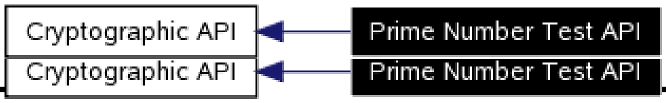
22 Prime Number Test API

23 素数测试API

[Cryptographic API]

[Cryptographic API]

Collaboration diagram for Prime Number Test API:
素数测试API的协作图:



23.1 Detailed Description

23.2 详细描述

File: cpa_cy_prime.h

文件:cpa_cy_prime.h

These functions specify the API for the prime number test operations.
这些函数指定了素数测试操作的API。

For prime number generation, this API SHOULD be used in conjunction with the Deterministic Random Bit Generation API (**Deterministic Random Bit Generation API**).
对于素数生成，这个API应该与确定性随机位生成API(Deterministic Random Bit Generation API

Note:
注意:

Large numbers are represented on the QuickAssist API as described in the Large Number API (**Cryptographic Large Number API**).
大数在QuickAssist API上表示，如大数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.
此外，对于椭圆曲线运算，传递给API的大数的位长度不得超过576位。

23.3 Data Structures

23.4 数据结构

- struct **_CpaCyPrimeTestOpData**

- 结构体_CpaCyPrimeTestOpData
- struct **_CpaCyPrimeStats**
- 结构体_CpaCyPrimeStats
- struct **_CpaCyPrimeStats64**
- 结构体_CpaCyPrimeStats64

23.5 Typedefs

23.6 类型定义

- typedef **_CpaCyPrimeTestOpData CpaCyPrimeTestOpData**

- 数据类型说明_CpaCyPrimeTestOpData CpaCyPrimeTestOpData
- typedef **_CpaCyPrimeStats CPA_DEPRECATED**
- 数据类型说明_CpaCyPrimeStats CPA_DEPRECATED
- typedef **_CpaCyPrimeStats64 CpaCyPrimeStats64**
- 数据类型说明_CpaCyPrimeStats64 CpaCyPrimeStats64
- typedef void(* **CpaCyPrimeTestCbFunc**)(void *pCallbackTag, **CpaStatus** status, void *pOpData,
- typedef void(*CpaCyPrimeTestCbFunc CpaStatus
  CpaBoolean testPassed)
  CpaBoolean 测试通过)

23.7 Functions

23.8 功能

- **CpaStatus cpaCyPrimeTest** (const **CpaInstanceHandle** instanceHandle, const **CpaCyPrimeTestCbFunc** pCb, void *pCallbackTag, const **CpaCyPrimeTestOpData** *pOpData, **CpaBoolean** *pTestPassed)

- CpaStatus cpaCyPrimeTest （常量CpaInstanceHandle CpaCyPrimeTestCbFunc CpaCyPrimeTestOpData CpaBoolean

23.9 Data Structure Documentation

23.10 数据结构文档

18.5 Data Structure Documentation

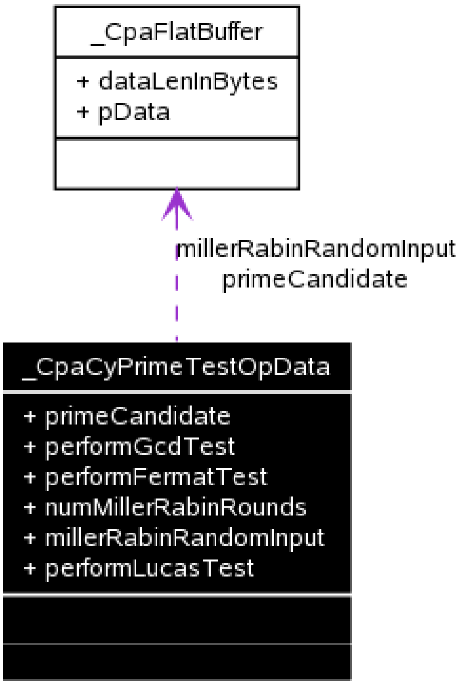
18.6 数据结构文档

18.6.1 _CpaCyPrimeTestOpData Struct Reference

18.6.2 _ CpaCyPrimeTestOpData 结构引用

Collaboration diagram for _CpaCyPrimeTestOpData:

_ CpaCyPrimeTestOpData 的协作图:



18.6.2.1 Detailed Description
18.6.2.2 详细描述

Prime Test Operation Data.
初始测试操作数据。

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.
此结构包含 cpaCyPrimeTest 函数的操作数据。客户端必须为这个结构和这个结构指向的项目分配内存。当结构被传递给函数时，内存的所有权就传递给了函数。当这个结构在回调函数中返回时，内存的所有权返回给客户端。

All values in this structure are required to be in Most Significant Byte first order, e.g. primeCandidate.pData[0]
此结构中的所有值都要求按最高有效字节优先顺序排列，例如 primeCandidate.pData[0]
= MSB.

= MSB。

All numbers MUST be stored in big-endian order.
所有数字都必须以大端顺序存储。

Note:
注意:

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.
如果客户端在将此结构中引用的内存提交给 cpaCyPrimeTest 函数之后，在回调中返回之前修改或释放该内存，将导致未定义的行为。

See also:
另请参见:

cpaCyPrimeTest()
cpaCyPrimeTest()

18.6.2.3 Data Fields
18.6.2.4 数据字段

- **CpaFlatBuffer primeCandidate**
- CpaFlatBuffer primeCandidate
- **CpaBoolean performGcdTest**
- CpaBoolean performGcdTest
- **CpaBoolean performFermatTest**
- CpaBoolean performFermatTest
- **Cpa32U numMillerRabinRounds**
- Cpa32U numMillerRabinRounds
- **CpaFlatBuffer millerRabinRandomInput**
- CpaFlatBuffer millerRabinRandomInput

18.5.1 _CpaCyPrimeTestOpData Struct Reference

18.5.2_ CpaCyPrimeTestOpData 结构引用

- **CpaBoolean performLucasTest**
- CpaBoolean performLucasTest

18.6.2.5 Field Documentation
18.6.2.6 现场文件

CpaFlatBuffer _CpaCyPrimeTestOpData::primeCandidate

The prime number candidate to test

要测试的质数候选

CpaBoolean _CpaCyPrimeTestOpData::performGcdTest

A value of CPA_TRUE means perform a GCD Primality Test

CPA_TRUE 的值意味着执行 GCD 素性测试

CpaBoolean _CpaCyPrimeTestOpData::performFermatTest

A value of CPA_TRUE means perform a Fermat Primality Test

CPA_TRUE 的值意味着执行费马素性测试

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.

米勒拉宾素性测试回合数。设置为 0 以执行零米勒拉宾测试。支持的最大回合数为 50。

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

包含一个指针的平面缓冲区，该指针指向用于 Miller Rabin 素性测试的 n 个随机数的数组。缓冲区的大小必须是

n * (MAX(64,x))

n *(最大值(64，x))

where:

其中:

- n is the requested number of rounds.
- n 是请求的回合数。
- x is the minimum number of bytes required to represent the prime candidate, i.e. x = ceiling((ceiling(log2(p)))/8).
- x 是表示主要候选所需的最小字节数，即 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.

每个随机数必须大于 1 且小于质数候选- 1，必要时以零开头。

CpaBoolean _CpaCyPrimeTestOpData::performLucasTest

An CPA_TRUE value means perform a Lucas Primality Test

CPA_TRUE 值意味着执行卢卡斯素性测试

18.5.3 _CpaCyPrimeStats Struct Reference

18.5.4 _CpaCyPrimeStats 结构引用

18.5.2.1 Detailed Description

Prime Number Test Statistics.

Deprecated:

18.5.2.2 素数检验统计量的详

细描述。**Deprecated:**

> As of v1.3 of the Crypto API, this structure has been deprecated, replaced by **CpaCyPrimeStats64**.
> 从 Crypto API 的 1.3 版开始，这种结构已被取代，由 **CpaCyPrimeStats64**

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.
这个结构包含素数测试操作的统计数据。当组件初始化时，统计信息被设置为零，并针对每个实例进行收集。

18.5.2.3 Data Fields
18.5.2.4 数据字段

- **Cpa32U numPrimeTestRequests**
- Cpa32U numPrimeTestRequests
- **Cpa32U numPrimeTestRequestErrors**
- Cpa32U numPrimeTestRequestErrors
- **Cpa32U numPrimeTestCompleted**
- Cpa32U numPrimeTestCompleted

18.5.2 _CpaCyPrimeStats Struct Reference
18.5.3_CpaCyPrimeStats 结构引用

- **Cpa32U numPrimeTestCompletedErrors**
- Cpa32U numPrimeTestCompletedErrors
- **Cpa32U numPrimeTestFailures**
- Cpa32U numPrimeTestFailures

18.5.2.5 Field Documentation
18.5.2.6 现场文件

Cpa32U _CpaCyPrimeStats::numPrimeTestRequests
Cpa32U _CpaCyPrimeStats::numPrimeTestRequests

Total number of successful prime number test requests.
成功的素数测试请求总数。

Cpa32U _CpaCyPrimeStats::numPrimeTestRequestErrors
Cpa32U _CpaCyPrimeStats::numPrimeTestRequestErrors

Total number of prime number test requests that had an error and could not be processed.
出错且无法处理的素数测试请求的总数。

Cpa32U _CpaCyPrimeStats::numPrimeTestCompleted
Cpa32U _CpaCyPrimeStats::numPrimeTestCompleted

Total number of prime number test operations that completed successfully.
成功完成的素数测试操作的总数。

Cpa32U _CpaCyPrimeStats::numPrimeTestCompletedErrors
Cpa32U _CpaCyPrimeStats::numPrimeTestCompletedErrors

Total number of prime number test operations that could not be completed successfully due to errors.
由于错误而无法成功完成的素数测试操作的总数。

Cpa32U _CpaCyPrimeStats::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.
成功执行的素数测试操作的总数，但测试结果显示该数字不是素数。

18.5.4 _CpaCyPrimeStats64 Struct Reference
18.5.5 _CpaCyPrimeStats64 结构引用

18.5.3.1 Detailed Description
18.5.3.2 详细描述

Prime Number Test Statistics (64-bit version).
素数测试统计(64 位版本)。

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.
此结构包含素数测试操作的 64 位版本的统计信息。当组件初始化时，统计信息被设置为零，并针对每个实例进行收集。

18.5.3.3 Data Fields
18.5.3.4 数据字段

- **Cpa64U numPrimeTestRequests**
- Cpa64U numPrimeTestRequests
- **Cpa64U numPrimeTestRequestErrors**
- Cpa64U numPrimeTestRequestErrors
- **Cpa64U numPrimeTestCompleted**
- Cpa64U numPrimeTestCompleted
- **Cpa64U numPrimeTestCompletedErrors**
- Cpa64U numPrimeTestCompletedErrors
- **Cpa64U numPrimeTestFailures**
- Cpa64U numPrimeTestFailures

18.5.3.5 Field Documentation
18.5.3.6 现场文件

Cpa64U _CpaCyPrimeStats64::numPrimeTestRequests

Cpa64U _CpaCyPrimeStats64::numPrimeTestRequests

Total number of successful prime number test requests.
成功的素数测试请求总数。

Cpa64U _CpaCyPrimeStats64::numPrimeTestRequestErrors

Cpa64U _CpaCyPrimeStats64::numPrimeTestRequestErrors

Total number of prime number test requests that had an error and could not be processed.
出错且无法处理的素数测试请求的总数。

Cpa64U _CpaCyPrimeStats64::numPrimeTestCompleted

Cpa64U _CpaCyPrimeStats64::numPrimeTestCompleted

Total number of prime number test operations that completed successfully.
成功完成的素数测试操作的总数。

Cpa64U _CpaCyPrimeStats64::numPrimeTestCompletedErrors

Cpa64U _CpaCyPrimeStats64::numPrimeTestCompletedErrors

Total number of prime number test operations that could not be completed successfully due to errors.

由于错误而无法成功完成的素数测试操作的总数。

Cpa64U _CpaCyPrimeStats64::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.
成功执行的素数测试操作的总数，但测试结果显示该数字不是素数。

18.7 Typedef Documentation

18.8 Typedef 文档

typedef struct **_CpaCyPrimeTestOpData CpaCyPrimeTestOpData**

typedef struct _CpaCyPrimeTestOpData CpaCyPrimeTestOpData

Prime Test Operation Data.
初始测试操作数据。

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.
此结构包含 cpaCyPrimeTest 函数的操作数据。客户端必须为这个结构和这个结构指向的项目分配内存。当结构被传递给函数时，内存的所有权就传递给了函数。当这个结构在回调函数中返回时，内存的所有权返回给客户端。

All values in this structure are required to be in Most Significant Byte first order, e.g. primeCandidate.pData[0] = MSB.
该结构中的所有值都要求按最高有效字节优先顺序排列，例如 primeCandidate.pData[0] = MSB。

All numbers MUST be stored in big-endian order.
所有数字都必须以大端顺序存储。

Note:
注意:

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.
如果客户端在将此结构中引用的内存提交给 cpaCyPrimeTest 函数之后，在回调中返回之前修改或释放该内存，将导致未定义的行为。

See also:
另请参见:

cpaCyPrimeTest()
cpaCyPrimeTest()

typedef struct **_CpaCyPrimeStats CPA_DEPRECATED**

typedef struct _CpaCyPrimeStats CPA_DEPRECATED

Prime Number Test Statistics.
素数检验统计。

Deprecated:

Deprecated:

As of v1.3 of the Crypto API, this structure has been deprecated, replaced by **CpaCyPrimeStats64**.
从 Crypto API 的 1.3 版开始，这种结构已被取代，由 **CpaCyPrimeStats64**

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.
这个结构包含素数测试操作的统计数据。当组件初始化时，统计信息被设置为零，并针对每个实例进行收集。

typedef struct _**CpaCyPrimeStats64 CpaCyPrimeStats64**

Prime Number Test Statistics (64-bit version).
素数测试统计(64 位版本)。

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.
此结构包含素数测试操作的 64 位版本的统计信息。当组件初始化时，统计信息被设置为零，并针对每个实例进行收集。

typedef void(* **CpaCyPrimeTestCbFunc**)(void *pCallbackTag, **CpaStatus** status, void *pOpData,

typedef void(* **CpaCyPrimeTestCbFunc**)(void *pCallbackTag, **CpaStatus** status, void *pOpData,

Definition of callback function invoked for cpaCyPrimeTest requests.

This is the prototype for the cpaCyPrimeTest callback function.

为 cpaCyPrimeTest 请求调用的回调函数的定义。这是

cpaCyPrimeTest 回调函数的原型。

Context:

背景:

This callback function can be executed in a context that DOES NOT permit sleeping to occur.
这个回调函数可以在不允许休眠发生的上下文中执行。

Assumptions:

假设:

None
没有人

Side-Effects:
副作用:

None
没有人

Reentrant:
可重入:

No
不

Thread-safe:
线程安全:

Yes
是

Parameters:
参数:

`[in]` *pCallbackTag* User-supplied value to help identify request.
[in] pCallbackTag 使用者提供的值，可协助识别要求。
`[in]` *status* Status of the operation. Valid values are CPA_STATUS_SUCCESS, CPA_STATUS_FAIL and CPA_STATUS_UNSUPPORTED.
[in] status 操作的状态。有效值为 CPA_STATUS_SUCCESS、CPA_STATUS_FAIL 和 CPA_STATUS_UNSUPPORTED。
`[in]` *pOpData* Opaque pointer to the Operation data pointer supplied in request.
[in]指向请求中提供的操作数据指针的 pOpData Opaque 指针。
`[in]` *testPassed* A value of CPA_TRUE means the prime candidate is probably prime.
[in] testPassed 值 CPA_TRUE 表示主候选项可能是主候选项。

Return values:
返回值:

None
没有人

Precondition:
前提条件:

Component has been initialized.
组件已初始化。

Postcondition:
后置条件:

None
没有人

Note:

注意:

None

没有人

See also:

另请参见:

cpaCyPrimeTest()

cpaCyPrimeTest()

18.8 Function Documentation

18.9 功能文档

```
cpaCyPrimeTest (  const                                           instanceHandle,
cpaCyPrimeTest  (  const                                          instanceHandle,
                        const pCb,
                        void *pCallbackTag,
                        const * pOpData,
                        *pTestPassed
                       )
```

Prime Number Test Function.

素数测试函数。

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.

这个函数将从概率上测试一个数是否是质数。详情请参考ANSI X9.80 2005。素性结果将在异步回调中返回。

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);

支持GCD、Fermat、Miller-Rabin和Lucas测试的以下组合: (高达1x GCD) +(高达1x Fermat) +(高达50x Miller-Rabin轮)+(高达1x Lucas)例如:(1x GCD)+(25x Miller-Rabin)+(1x Lucas); (1x GCD)+(1x Fermat); (50x米勒-拉宾);

Tests are always performed in order of increasing complexity, for example GCD first, then Fermat, then Miller-Rabin, and finally Lucas.

测试总是按照复杂性增加的顺序进行，例如首先是GCD，然后是Fermat，然后是Miller-Rabin，最后是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.

对于所有的素性测试，支持以下素数“大小”(以位为单位的长度):512 位以下的所有大小，以及大小 768、1024、1536、2048、3072 和 4096。

Candidate prime numbers MUST match these sizes accordingly, with leading zeroes present where necessary.
候选质数必须相应地匹配这些大小，必要时在前面加零。

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.
当这个素数测试与米勒-拉宾和卢卡斯组合测试一起使用时，它可以被用作在随机数据发生器上执行自测操作的一种手段。

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).
ok (pass == CPA_TRUE)的响应状态意味着通过了所有请求的素性测试，并且素候选可能是素的(确切的概率取决于请求的素性测试)。不正常的响应状态(pass == CPA_FALSE)意味着所请求的素性测试之一失败(已发现素候选是复合的)。

Context:
背景:

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.
当作为异步函数调用时，它不能休眠。它可以在不允许休眠的上下文中执行。当作为同步函数调用时，它可能会休眠。它不能在不允许休眠的上下文中执行。

Assumptions:
假设:

None
没有人

Side-Effects:
副作用:

None
没有人

Blocking:
阻止:

Yes when configured to operate in synchronous mode.
当配置为在同步模式下运行时，是。

Reentrant:
可重入:

No
不

Thread-safe:

线程安全:

Yes

是

Parameters:

参数:

[in] *instanceHandle* Instance handle.

[in] instanceHandle 执行个体控制代码。

[in] *pCb* Callback function pointer. If this is set to a NULL value the function will operate synchronously.

[in] pCb 回调函数指针。如果设置为空值，函数将同步运行。

[in] *pCallbackTag* User-supplied value to help identify request.

[in] pCallbackTag 使用者提供的值，可协助识别要求。

[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.

[in] pOpData 结构，包含执行作业所需的所有资料。客户端代码为此结构分配内存。该组件取得内存的所有权，直到它在回调中被返回。

[out] *pTestPassed* A value of CPA_TRUE means the prime candidate is probably prime.

[out] pTestPassed 值 CPA_TRUE 表示主要候选项可能是主要的。

Return values:

返回值:

CPA_STATUS_SUCCESS Function executed successfully.

CPA_STATUS_SUCCESS 函数执行成功。

CPA_STATUS_FAIL Function failed.

CPA_STATUS_FAIL 函数失败。

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.

CPA_STATUS_RETRY 重新提交请求。传递的 CPA_STATUS_INVALID_PARAM 参数无效。与系统资源相关的 CPA_STATUS_RESOURCE 错误。CPA_STATUS_RESTARTING API 实现正在重新启动。重新提交请求。不支持 CPA_STATUS_UNSUPPORTED 函数。

Precondition:

前提条件:

The component has been initialized via cpaCyStartInstance function.

该组件已通过 cpaCyStartInstance 函数初始化。

Postcondition:

后置条件:

None

没有人

Note:

注意:

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.
当 pCb 为非空时，会生成一个 CpaCyPrimeTestCbFunc 类型的异步回调来响应此函数调用。为了获得最佳性能，数据指针应该 8 字节对齐。

See also:

另请参见:

CpaCyPrimeTestOpData, **CpaCyPrimeTestCbFunc**
CpaCyPrimeTestOpData, CpaCyPrimeTestCbFunc

24 Deterministic Random Bit Generation API

25 确定性随机位生成API

[Cryptographic API]

[Cryptographic API]

Collaboration diagram for Deterministic Random Bit Generation API:
确定性随机位生成API的协作图:

Cryptographic API
Deterministic Random Bit Generation API
Cryptographic API
Deterministic Random Bit Generation API

25.1 Detailed Description

25.2 详细描述

File: cpa_cy_drbg.h

文件:cpa_cy_drbg.h

These functions specify the API for a Deterministic Random Bit Generation (DRBG), compliant with NIST SP 800-90, March 2007, "Recommendation for Random Number Generation Using Deterministic Random Bit Generators (Revised)".
这些函数指定了用于确定性随机位生成(DRBG)的API，符合NIST SP 800-90，2007年3月，“使用确定性随机位生成器生成随机数的建议(修订版)”。

The functions **cpaCyDrbgInitSession**, **cpaCyDrbgGen**, **cpaCyDrbgReseed** and **cpaCyDrbgRemoveSession** are used to instantiate, generate, reseed and uninstantiate a DRBG mechanism.
这些功能cpaCyDrbgInitSessioncpaCyDrbgGencpaCyDrbgReseed cpaCyDrbgRemoveSession

Note:
注意:

> These functions supersede the random number generation functions in API group **Random Bit/Number Generation API**, which are now deprecated.
> 这些函数取代了API组中的随机数生成函数RandomBit/Number Generation API

25.3 Data Structures

25.4 数据结构

- struct **_CpaCyDrbgSessionSetupData**

- 结构体_CpaCyDrbgSessionSetupData
- struct **_CpaCyDrbgGenOpData**
- 结构体_CpaCyDrbgGenOpData
- struct **_CpaCyDrbgReseedOpData**
- 结构体_CpaCyDrbgReseedOpData
- struct **_CpaCyDrbgStats64**
- 结构体_CpaCyDrbgStats64

25.5 Typedefs
25.6 类型定义

- typedef enum **_CpaCyDrbgSecStrength CpaCyDrbgSecStrength**

- typedef 枚举_CpaCyDrbgSecStrength CpaCyDrbgSecStrength
- typedef **_CpaCyDrbgSessionSetupData CpaCyDrbgSessionSetupData**
- 数据类型说明_CpaCyDrbgSessionSetupData CpaCyDrbgSessionSetupData
- typedef void * **CpaCyDrbgSessionHandle**
- typedef void *CpaCyDrbgSessionHandle
- typedef **_CpaCyDrbgGenOpData CpaCyDrbgGenOpData**
- 数据类型说明_CpaCyDrbgGenOpData CpaCyDrbgGenOpData
- typedef **_CpaCyDrbgReseedOpData CpaCyDrbgReseedOpData**
- 数据类型说明_CpaCyDrbgReseedOpData CpaCyDrbgReseedOpData
- typedef **_CpaCyDrbgStats64 CpaCyDrbgStats64**
- 数据类型说明_CpaCyDrbgStats64 CpaCyDrbgStats64

25.7 Enumerations
25.8 列举

- enum **_CpaCyDrbgSecStrength** {
  CPA_CY_RBG_SEC_STRENGTH_112,
  CPA_CY_RBG_SEC_STRENGTH_128,
  CPA_CY_RBG_SEC_STRENGTH_192,
  CPA_CY_RBG_SEC_STRENGTH_256

- 列举型别_CpaCyDrbgSecStrength
  }
  }

25.9 Functions

25.10 功能

19.5 Functions

19.6 功能

- **CpaStatus cpaCyDrbgSessionGetSize** (const **CpaInstanceHandle** instanceHandle, const

- CpaStatus cpaCyDrbgSessionGetSize（常量 CpaInstanceHandle
  CpaCyDrbgSessionSetupData *pSetupData, **Cpa32U** *pSize)
  CpaCyDrbgSessionSetupData *pSetupData，Cpa32U
- **CpaStatus cpaCyDrbgInitSession** (const **CpaInstanceHandle** instanceHandle, const **CpaCyGenFlatBufCbFunc** pGenCb, const **CpaCyGenericCbFunc** pReseedCb, const **CpaCyDrbgSessionSetupData** *pSetupData, **CpaCyDrbgSessionHandle** sessionHandle, **Cpa32U**
- CpaStatus cpaCyDrbgInitSession（常量 CpaInstanceHandle CpaCyGenFlatBufCbFunc CpaCyGenericCbFunc CpaCyDrbgSessionSetupData CpaCyDrbgSessionHandle Cpa32U
  *pSeedLen)
  *pSeedLen）
- **CpaStatus cpaCyDrbgReseed** (const **CpaInstanceHandle** instanceHandle, void *pCallbackTag,
- CpaStatus cpaCyDrbgReseed（常量 CpaInstanceHandle
  CpaCyDrbgReseedOpData *pOpData)
  CpaCyDrbgReseedOpData *pOpData）
- **CpaStatus cpaCyDrbgGen** (const **CpaInstanceHandle** instanceHandle, void *pCallbackTag,
- CpaStatus cpaCyDrbgGen（常量 CpaInstanceHandle
  CpaCyDrbgGenOpData *pOpData, **CpaFlatBuffer** *pPseudoRandomBits)
  CpaCyDrbgGenOpData *pOpData，CpaFlatBuffer
- **CpaStatus cpaCyDrbgRemoveSession** (const **CpaInstanceHandle** instanceHandle,
- CpaStatus cpaCyDrbgRemoveSession（常量 CpaInstanceHandle
  CpaCyDrbgSessionHandle sessionHandle)
  CpaCyDrbgSessionHandle sessionHandle）
- **CpaStatus cpaCyDrbgQueryStats64** (const **CpaInstanceHandle** instanceHandle,
- CpaStatus cpaCyDrbgQueryStats64（常量 CpaInstanceHandle
  CpaCyDrbgStats64 *pStats)
  CpaCyDrbgStats64 * pStats）

19.7 Data Structure Documentation

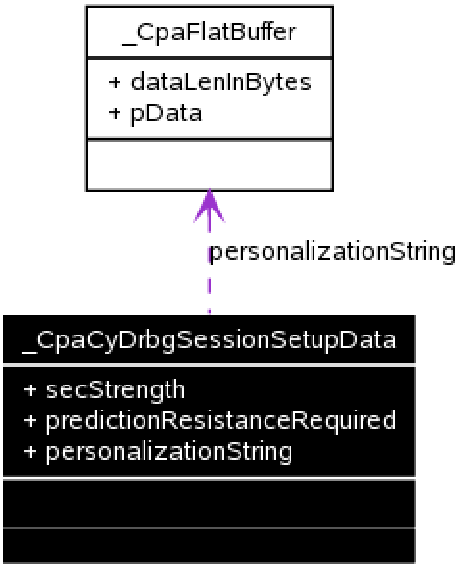
19.8 数据结构文档

19.8.1 _CpaCyDrbgSessionSetupData Struct Reference

19.8.2 _CpaCyDrbgSessionSetupData 结构引用

Collaboration diagram for _CpaCyDrbgSessionSetupData:

_CpaCyDrbgSessionSetupData 的协作图:



19.8.2.1 Detailed Description
19.8.2.2 详细描述

DRBG Session (Instance) Setup Data
DRBG 会话(实例)设置数据

This structure contains data relating to instantiation of a DRBG session, or instance.
该结构包含与 DRBG 会话或实例的实例化相关的数据。

19.8.2.3 Data Fields
19.8.2.4 数据字段

- **CpaCyDrbgSecStrength secStrength**
- CpaCyDrbgSecStrength secStrength
- **CpaBoolean predictionResistanceRequired**
- CpaBoolean predictionResistanceRequired
- **CpaFlatBuffer personalizationString**
- CpaFlatBuffer personalizationString

19.6.1 _CpaCyDrbgSessionSetupData Struct Reference

19.6.2_CpaCyDrbgSessionSetupData 结构引用

19.8.2.5 Field Documentation

19.8.2.6 现场文件

CpaCyDrbgSecStrength _CpaCyDrbgSessionSetupData::secStrength

Requested security strength
要求的安全力量

CpaBoolean _CpaCyDrbgSessionSetupData::predictionResistanceRequired

Prediction resistance flag. Indicates whether or not prediction resistance may be required by the consuming application during one or more requests for pseudorandom bits.
预测阻力标志。指示消费应用程序在一个或多个伪随机位请求期间是否需要预测阻力。

CpaFlatBuffer _CpaCyDrbgSessionSetupData::personalizationString

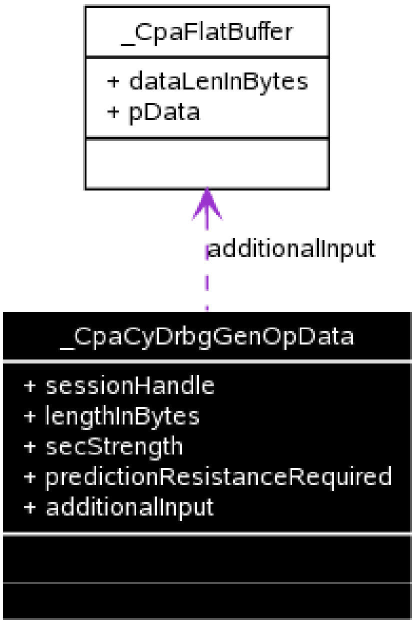
Personalization string. String that should be used to derive the seed.
个性化字符串。应该用于派生种子的字符串。

19.6.3 _CpaCyDrbgGenOpData Struct Reference

19.6.4 _CpaCyDrbgGenOpData 结构引用

Collaboration diagram for _CpaCyDrbgGenOpData:

_CpaCyDrbgGenOpData 的协作图:



19.6.4.1 Detailed Description

19.6.4.2 详细描述

DRBG Data Generation Operation Data
DRBG 数据生成操作数据

This structure contains data relating to generation of random bits using a DRBG.
该结构包含与使用 DRBG 生成随机位相关的数据。

See also:
另请参见：

cpaCyDrbgGen()
cpaCyDrbgGen()

Note:

注意：

提交给 cpaCyDrbgGen()

If the client modifies or frees the memory referenced in this structure after it has been submitted to the **cpaCyDrbgGen()** function, and before it has been returned in the callback, undefined behavior will result. 如果客户端在将此结构

19.6.2 _CpaCyDrbgGenOpData Struct Reference

19.6.3_CpaCyDrbgGenOpData 结构引用

19.6.4.3 Data Fields

19.6.4.4 数据字段

- **CpaCyDrbgSessionHandle sessionHandle**
- CpaCyDrbgSessionHandle sessionHandle
- **Cpa32U lengthInBytes**
- Cpa32U lengthInBytes
- **CpaCyDrbgSecStrength secStrength**
- CpaCyDrbgSecStrength secStrength
- **CpaBoolean predictionResistanceRequired**
- CpaBoolean predictionResistanceRequired
- **CpaFlatBuffer additionalInput**
- CpaFlatBuffer additionalInput

19.6.4.5 Field Documentation
19.6.4.6 现场文件

CpaCyDrbgSessionHandle _CpaCyDrbgGenOpData::sessionHandle

Session handle, also known as the state handle or instance handle
会话句柄，也称为状态句柄或实例句柄

Cpa32U _CpaCyDrbgGenOpData::lengthInBytes

Requested number of bytes to be generated
请求生成的字节数

CpaCyDrbgSecStrength _CpaCyDrbgGenOpData::secStrength

Requested security strength
要求的安全力量

CpaBoolean _CpaCyDrbgGenOpData::predictionResistanceRequired

Requested prediction resistance flag. Indicates whether or not prediction resistance is to be provided prior to the generation of the requested pseudorandom bits to be generated.
请求的预测阻力标志。指示在生成要生成的所请求的伪随机位之前是否要提供预测阻力。

CpaFlatBuffer _CpaCyDrbgGenOpData::additionalInput

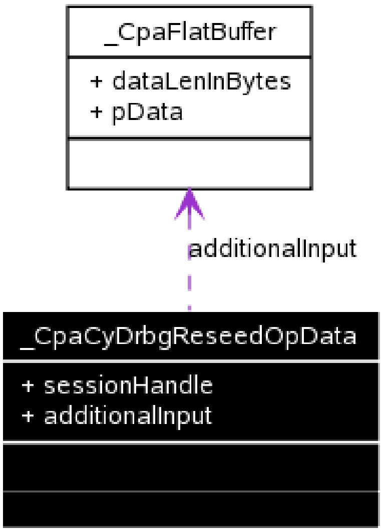
Additional input
附加输入

19.6.4 _CpaCyDrbgReseedOpData Struct Reference

19.6.5 _CpaCyDrbgReseedOpData 结构引用

Collaboration diagram for _CpaCyDrbgReseedOpData:

_CpaCyDrbgReseedOpData 的协作图:



19.6.5.1 Detailed Description
19.6.5.2 详细描述

DRBG Reseed Operation Data
DRBG 重新播种操作数据

This structure contains data relating to reseeding a DRBG session, or instance.
此结构包含与重新播种 DRBG 会话或实例相关的数据。

19.6.3 _CpaCyDrbgReseedOpData Struct Reference

19.6.4_CpaCyDrbgReseedOpData 结构引用

See also:

另请参见:

cpaCyDrbgReseed()

cpaCyDrbgReseed()

Note:

注意:

If the client modifies or frees the memory referenced in this structure after it has been submitted to the **cpaCyDrbgReseed()** function, and before it has been returned in the callback, undefined behavior will result.
如果客户端在将此结构提交给 cpaCyDrbgReseed()

19.6.5.3 Data Fields

19.6.5.4 数据字段

- **CpaCyDrbgSessionHandle sessionHandle**
- CpaCyDrbgSessionHandle sessionHandle
- **CpaFlatBuffer additionalInput**
- CpaFlatBuffer additionalInput

19.6.5.5 Field Documentation
19.6.5.6 现场文件

CpaCyDrbgSessionHandle _CpaCyDrbgReseedOpData::sessionHandle

CpaCyDrbgSessionHandle _CpaCyDrbgReseedOpData::sessionHandle

Session handle, also known as a state handle or instance handle.
会话句柄，也称为状态句柄或实例句柄。

CpaFlatBuffer _CpaCyDrbgReseedOpData::additionalInput

CpaFlatBuffer _CpaCyDrbgReseedOpData::additionalInput

An "optional" input to the reseeding. The length should be less than or equal to the seed length, which is returned by the function **cpaCyDrbgInitSession()**. A length of 0 can be specified to indicate no additional input.
补种的“可选”输入。长度应该小于或等于种子长度，种子长度由函数返回 cpaCyDrbgInitSession()

19.6.5 _CpaCyDrbgStats64 Struct Reference

19.6.6 _CpaCyDrbgStats64 结构引用

19.6.6.1 Detailed Description
19.6.6.2 详细描述

DRBG Statistics
DRBG 统计

This structure contains statistics (counters) related to the random bit generation API.
该结构包含与随机位生成 API 相关的统计信息(计数器)。

See also:
另请参见:

CpaCyDrbgQueryStats64()
CpaCyDrbgQueryStats64()

19.6.6.3 Data Fields
19.6.6.4 数据字段

- **Cpa64U numSessionsInitialized**
- Cpa64U numSessionsInitialized
- **Cpa64U numSessionsRemoved**
- Cpa64U numSessionsRemoved
- **Cpa64U numSessionErrors**
- Cpa64U numSessionErrors
- **Cpa64U numGenRequests**
- Cpa64U numGenRequests
- **Cpa64U numGenRequestErrors**
- Cpa64U numGenRequestErrors
- **Cpa64U numGenCompleted**
- Cpa64U numGenCompleted
- **Cpa64U numGenCompletedErrors**
- Cpa64U numGenCompletedErrors
- **Cpa64U numReseedRequests**
- Cpa64U numReseedRequests
- **Cpa64U numReseedRequestErrors**
- Cpa64U numReseedRequestErrors
- **Cpa64U numReseedCompleted**
- Cpa64U numReseedCompleted
- **Cpa64U numReseedCompletedErrors**
- Cpa64U numReseedCompletedErrors

19.6.6.5 Field Documentation

19.6.6.6 现场文件

Cpa64U CpaCyDrbgStats64::numSessionsInitialized

Cpa64U CpaCyDrbgStats64::numSessionsInitialized

Number of session initialized

初始化的会话数

19.6.4 _CpaCyDrbgStats64 Struct Reference

CpaCyDrbgStats64 结构参考

Cpa64U _CpaCyDrbgStats64::numSessionsRemoved

Number of sessions removed
删除的会话数

Cpa64U _CpaCyDrbgStats64::numSessionErrors

Total number of errors returned when initializing and removing sessions
初始化和删除会话时返回的错误总数

Cpa64U _CpaCyDrbgStats64::numGenRequests

Number of successful calls to **cpaCyDrbgGen**.
成功调用的次数 cpaCyDrbgGen

Cpa64U _CpaCyDrbgStats64::numGenRequestErrors

Number of calls to **cpaCyDrbgGen** that returned an error and could not be processed.
呼叫次数 cpaCyDrbgGen

Cpa64U _CpaCyDrbgStats64::numGenCompleted

Number of calls to **cpaCyDrbgGen** that completed successfully.
呼叫次数 cpaCyDrbgGen

Cpa64U _CpaCyDrbgStats64::numGenCompletedErrors

Number of calls to **cpaCyDrbgGen** that completed with an error status.
呼叫次数 cpaCyDrbgGen

Cpa64U _CpaCyDrbgStats64::numReseedRequests

Number of successful calls to **cpaCyDrbgReseed**.
成功调用的次数 cpaCyDrbgReseed

Note that this does NOT include implicit reseeds due to calls to **cpaCyDrbgGen** with prediction resistance, or due to seed lifetime expiry.
请注意，这不包括由于调用 cpaCyDrbgGen

Cpa64U _CpaCyDrbgStats64::numReseedRequestErrors

Number of calls to **cpaCyDrbgReseed** that returned an error and could not be processed.
呼叫次数 cpaCyDrbgReseed

Cpa64U _CpaCyDrbgStats64::numReseedCompleted

Number of calls to **cpaCyDrbgReseed** that completed successfully.
呼叫次数 cpaCyDrbgReseed

Cpa64U _CpaCyDrbgStats64::numReseedCompletedErrors

Number of calls to **cpaCyDrbgReseed** that completed with an error status.
呼叫次数 cpaCyDrbgReseed

19.9 Typedef Documentation

19.10 Typedef 文档

typedef enum **_CpaCyDrbgSecStrength CpaCyDrbgSecStrength**

Security Strength
安全强度

This enum defines the security strength. NIST SP 800-90 defines security strength as "A number

associated with the amount of work (that is, the number of operations) that is required to break a cryptographic algorithm or system; a security strength is specified in bits and is a specific value from the set (112, 128, 192, 256) for this Recommendation. The amount of work needed is 2^(security_strength)."
此枚举定义安全强度。NIST SP 800-90 将安全强度定义为“与破解加密算法或系统所需的工作量(即操作次数)相关的数字；安全强度以比特为单位指定，是本建议集合(112，128，192，256)中的一个特定值。需要的工作量是 2^(security_strength).”

typedef struct _CpaCyDrbgSessionSetupData CpaCyDrbgSessionSetupData

DRBG Session (Instance) Setup Data
DRBG 会话(实例)设置数据

This structure contains data relating to instantiation of a DRBG session, or instance.
该结构包含与 DRBG 会话或实例的实例化相关的数据。

typedef void* CpaCyDrbgSessionHandle

Handle to a DRBG session (or instance).
DRBG 会话(或实例)的句柄。

19.7 Typedef Documentation

19.8 Typedef 文档

This is what NIST SP 800-90 refers to as the "state_handle". That document also refers to the process of creating such a handle as "instantiation", or instance creation. On this API, we use the term "session" to refer to such an instance, to avoid confusion with the crypto instance handle, and for consistency with the similar concept of sessions in symmetric crypto (see **Symmetric Cipher and Hash Cryptographic API**) and elsewhere on the API.

这就是NIST SP 800-90所称的“状态句柄”。该文档还将创建这种句柄的过程称为“实例化”或实例创建。在此API中，我们使用术语“会话”来指代此类实例，以避免与加密实例句柄混淆，并与对称加密中的类似会话概念保持一致(请参见**Symmetric Cipher and Hash Cryptographic API**

Note that there can be multiple sessions, or DRBG instances, created within a single instance of a CpaInstanceHandle.
请注意，在一个CpaInstanceHandle实例中可以创建多个会话或DRBG实例。

Note:
注意:

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 **cpaCyDrbgSessionGetSize** function. The session memory is initialized with a call to the **cpaCyDrbgInitSession** function. This memory MUST not be freed until a call to **cpaCyDrbgRemoveSession** has completed successfully.
此句柄的内存由客户端分配。客户端需要分配的内存大小是通过调用**cpaCyDrbgSessionGetSize** **cpaCyDrbgInitSession** **cpaCyDrbgRemoveSession**

typedef struct _**CpaCyDrbgGenOpData CpaCyDrbgGenOpData**

DRBG Data Generation Operation Data
DRBG数据生成操作数据

This structure contains data relating to generation of random bits using a DRBG.
该结构包含与使用DRBG生成随机位相关的数据。

See also:
另请参见:

cpaCyDrbgGen()
cpaCyDrbgGen()

Note:

注意:

If the client modifies or frees the memory referenced in this structure after it has been submitted to the **cpaCyDrbgGen()** function, and before it has been returned in the callback, undefined behavior will result.
如果客户端在将此结构提交给 **cpaCyDrbgGen()**

typedef struct _**CpaCyDrbgReseedOpData CpaCyDrbgReseedOpData**

typedef struct _CpaCyDrbgReseedOpData CpaCyDrbgReseedOpData

DRBG Reseed Operation Data
DRBG 重新播种操作数据

This structure contains data relating to reseeding a DRBG session, or instance.
此结构包含与重新播种 DRBG 会话或实例相关的数据。

See also:
另请参见:

cpaCyDrbgReseed()
cpaCyDrbgReseed()

Note:

注意:

If the client modifies or frees the memory referenced in this structure after it has been submitted to the **cpaCyDrbgReseed()** function, and before it has been returned in the callback, undefined behavior will result.
如果客户端在将此结构提交给 **cpaCyDrbgReseed()**

typedef struct _CpaCyDrbgStats64 **CpaCyDrbgStats64**

typedef struct _CpaCyDrbgStats64 CpaCyDrbgStats64

DRBG Statistics
DRBG 统计

This structure contains statistics (counters) related to the random bit generation API.
该结构包含与随机位生成 API 相关的统计信息(计数器)。

See also:
另请参见:

CpaCyDrbgQueryStats64()
CpaCyDrbgQueryStats64()

19.9 Enumeration Type Documentation

19.10 枚举类型文档

enum _CpaCyDrbgSecStrength

CpaCyDrbgSecStrength

Security Strength
安全强度

19.8 Enumeration Type Documentation

19.9 枚举类型文档

This enum defines the security strength. NIST SP 800-90 defines security strength as "A number associated with the amount of work (that is, the number of operations) that is required to break a cryptographic algorithm or system; a security strength is specified in bits and is a specific value from the set (112, 128, 192, 256) for this Recommendation. The amount of work needed is 2^(security_strength)."

此枚举定义安全强度。NIST SP 800-90 将安全强度定义为“与破解加密算法或系统所需的工作量(即操作次数)相关的数字；安全强度以比特为单位指定，是本建议集合(112，128，192，256)中的一个特定值。需要的工作量是 2^(security_strength).”

19.10 Function Documentation

19.11 功能文档

cpaCyDrbgSessionGetSize (const instanceHandle,

```
cpaCyDrbgSessionGetSize  (  const                                        instanceHandle,
                                 const *  pSetupData,
                                 *pSize
                               )
```

Returns the size (in bytes) of a DRBG session handle.
返回 DRBG 会话句柄的大小(以字节为单位)。

This function is used by the client to determine the size of the memory it must allocate in order to store the DRBG session. This MUST be called before the client allocates the memory for the session and before the client calls the **cpaCyDrbgInitSession** function.
客户端使用此函数来确定它必须分配的内存大小，以便存储 DRBG 会话。必须在客户端为会话分配内存之前以及客户端调用 **cpaCyDrbgInitSession**

Context:
背景:

This is a synchronous function and it can sleep. It MUST NOT be executed in a context that DOES NOT permit sleeping.
这是一个同步功能，它可以休眠。它不能在不允许休眠的上下文中执行。

Assumptions:
假设:

None
没有人

Side-Effects:
副作用:

None
没有人

Blocking:
阻止:

No.
号码

Reentrant:
可重入:

No
不

Thread-safe:
线程安全:

Yes
是

Parameters:
参数:

`[in]` *instanceHandle* Instance handle.
[in] instanceHandle 执行个体控制代码。
`[in]` *pSetupData* Pointer to session setup data which contains parameters which are static for a given DRBG session, such as security strength, etc.
[in] pSetupData 指向会话设置数据的指针，该数据包含给定 DRBG 会话的静态参数，如安全强度等。
`[out]` *pSize* The amount of memory in bytes required to hold the session.
[out] pSize 保存会话所需的内存量(以字节为单位)。

Return values:
返回值:

CPA_STATUS_SUCCESS Function executed successfully.
CPA_STATUS_SUCCESS 函数执行成功。
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.

CPA_STATUS_FAIL 函数失败。传递的 CPA_STATUS_INVALID_PARAM 参数无效。与系统资源相关的 CPA_STATUS_RESOURCE 错误。不支持 CPA_STATUS_UNSUPPORTED 函数。

Precondition:

前提条件:

The component has been initialized via the **cpaCyStartInstance** function.
该组件已通过 **cpaCyStartInstance**

Postcondition:
后置条件:

None

没有人

cpaCyDrbgInitSession(const instanceHandle

```
cpaCyDrbgInitSession(const                                       instanceHandle,
                              const pGenCb, const pReseedCb,  const


                              * *                                pSetupData,
                                                                 sessionHandle, pSeedLen


                            )
```

Instantiates and seeds a DRBG session, or instance.
实例化和播种 DRBG 会话或实例。

This function is used by the client to initialize a DRBG session, or instance.
客户端使用此函数来初始化 DRBG 会话或实例。

Note:
注意:

On some implementations, the client may have to register an entropy source, nonce source, and/or a function which specifies whether a derivation function is required. See the Programmer's Guide for your implementation for more details.
在一些实施方式中，客户端可能必须注册熵源、随机数源和/或指定是否需要推导函数的函数。有关更多详细信息，请参见您的实现的程序员指南。

Context:
背景:

This is a synchronous function and it can sleep. It MUST NOT be executed in a context that DOES NOT permit sleeping.
这是一个同步功能，它可以休眠。它不能在不允许休眠的上下文中执行。

Assumptions:
假设:

None
没有人

Side-Effects:
副作用:

None
没有人

Blocking:
阻止:

No.
号码

Reentrant:
可重入:

No
不

Thread-safe:
线程安全:

Yes
是

Parameters:
参数:

`[in]` *instanceHandle* Instance handle.

[in] instanceHandle 执行个体控制代码。

`[in]` *pGenCb* Pointer to callback function to be registered. This is the function that will be called back to indicate completion of the asynchronous **cpaCyDrbgGen** function. Set this field to NULL if this function is to operate in a synchronous manner.

[in]指向要注册的回调函数的 pGenCb 指针。这是将被回调以指示异步完成的函数 **cpaCyDrbgGen**

`[in]` *pReseedCb* Pointer to callback function to be registered. This is the function that will be called back to indicate completion of the asynchronous **cpaCyDrbgReseed** function. Set this field to NULL if this function is to operate in a synchronous manner.

[in]指向要注册的回调函数的 pReseedCb 指针。这是将被回调以指示异步完成的函数 **cpaCyDrbgReseed**

`[in]` *pSetupData* Pointer to setup data.

[in]指向安装数据的 pSetupData 指针。

`[out]` *sessionHandle* Pointer to the memory allocated by the client to store the instance handle. This will be initialized with this function. This handle needs to be passed to subsequent processing calls.

[out] sessionHandle 指向客户端为存储实例句柄而分配的内存的指针。这将用这个函数来初始化。这个句柄需要传递给后续的处理调用。

`[out]` *pSeedLen* Seed length for the supported DRBG mechanism and security strength. The value of this is dependent on the DRBG mechanism implemented by the instance, which is implementation-dependent. This seed length may

[out]受支持的 DRBG 机制和安全强度的 pSeedLen 种子长度。其值取决于实例实现的 DRBG 机制，而 DRBG 机制是依赖于实现的。这个种子长度可以

be used by the client when reseeding.

由客户在补种时使用。

Return values:
返回值:

CPA_STATUS_SUCCESS Function executed successfully.
CPA_STATUS_SUCCESS 函数执行成功。
CPA_STATUS_FAIL Function failed.
CPA_STATUS_FAIL 函数失败。
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.

CPA_STATUS_RETRY 重新提交请求。传递的 CPA_STATUS_INVALID_PARAM 参数无效。与系统资源相关的 CPA_STATUS_RESOURCE 错误。CPA_STATUS_RESTARTING API 实现正在重新启动。重新提交请求。不支持 CPA_STATUS_UNSUPPORTED 函数。

Precondition:

前提条件:

The component has been initialized via the **cpaCyStartInstance** function.
该组件已通过 cpaCyStartInstance

Postcondition:
后置条件:

None
没有人

cpaCyDrbgReseed (const instanceHandle

```
cpaCyDrbgReseed (  const    instanceHandle,
void *pCallbackTag,
                               pOpData
                               *
                             )
```

Reseeds a DRBG session, or instance.
为 DRBG 会话或实例重新设定种子。

Reseeding inserts additional entropy into the generation of pseudorandom bits.
重新播种将额外的熵插入到伪随机位的生成中。

Context:
背景:

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.
当作为异步函数调用时，它不能休眠。它可以在不允许休眠的上下文中执行。当作为同步函数调用时，它可能会休眠。它不能在不允许休眠的上下文中执行。

Assumptions:
假设:

None
没有人

Side-Effects:
副作用:

None
没有人

Blocking:
阻止:

Yes when configured to operate in synchronous mode.
当配置为在同步模式下运行时，是。

Reentrant:
可重入:

No
不

Thread-safe:
线程安全:

Yes
是

Parameters:
参数:

`[in]` *instanceHandle* Instance handle.
[in] instanceHandle 执行个体控制代码。

`[in]` *pCallbackTag* Opaque User Data for this specific call. Will be returned unchanged in the callback.
[in] pCallbackTag 此特定调用的不透明用户数据。将在回调中不变地返回。

`[in]` *pOpData* Structure containing all the data needed to perform the operation. The client code allocates the memory for this structure.
[in] pOpData 结构，包含执行作业所需的所有资料。客户端代码为此结构分配内存。

Return values:

返回值:

CPA_STATUS_SUCCESS Function executed successfully.
CPA_STATUS_SUCCESS 函数执行成功。

CPA_STATUS_FAIL Function failed.

CPA_STATUS_FAIL 函数失败。

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.

CPA_STATUS_RETRY 重新提交请求。传递的 CPA_STATUS_INVALID_PARAM 参数无效。与系统资源相关的 CPA_STATUS_RESOURCE 错误。CPA_STATUS_RESTARTING API 实现正在重新启动。重新提交请求。不支持 CPA_STATUS_UNSUPPORTED 函数。

Precondition:

前提条件:

The component has been initialized via the **cpaCyStartInstance** function.
该组件已通过 cpaCyStartInstance

Postcondition:
后置条件:

None
没有人

```
CpaStatus cpaCyDrbgGen ( const CpaInstanceHandle instanceHandle,
                         void *                  pCallbackTag,
                         CpaCyDrbgGenOpData *    pOpData,
                         CpaFlatBuffer *         pPseudoRandomBits
                       )
```

```
CpaStatus cpaCyDrbgGen ( const CpaInstanceHandle instanceHandle,
                         void *                  pCallbackTag,
                         CpaCyDrbgGenOpData *    pOpData,
                         CpaFlatBuffer *         pPseudoRandomBits
                       )
```

Generates pseudorandom bits.
产生伪随机位。

This function is used to request the generation of random bits. The generated data and the length of the data will be returned to the caller in an asynchronous callback function.
该函数用于请求生成随机位。生成的数据和数据长度将在异步回调函数中返回给调用者。

Context:
背景:

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.
当作为异步函数调用时，它不能休眠。它可以在不允许休眠的上下文中执行。当作为同步函数调用时，它可能会休眠。它不能在不允许休眠的上下文中执行。

Assumptions:
假设:

None
没有人

Side-Effects:
副作用:

None
没有人

Blocking:
阻止:

Yes when configured to operate in synchronous mode.
当配置为在同步模式下运行时，是。

Reentrant:
可重入:

No
不

Thread-safe:
线程安全:

Yes
是

Parameters:
参数:

`[in]` *instanceHandle* Instance handle.
[in] instanceHandle 执行个体控制代码。

`[in]` *pCallbackTag* Opaque User Data for this specific call. Will be returned unchanged
[in] pCallbackTag 此特定调用的不透明用户数据。将原封不动地退回
in the callback.
在回调中。

`[in]` *pOpData* Structure containing all the data needed to perform the operation.
[in] pOpData 结构，包含执行作业所需的所有资料。
The client code allocates the memory for this structure. This component takes ownership of the memory until it is returned in the callback.
客户端代码为此结构分配内存。该组件取得内存的所有权，直到它在回调中被返回。

`[out]` *pPseudoRandomBits* Pointer to the memory allocated by the client where the random data
[out] pPseudoRandomBits 指向由客户端分配的内存的指针
will be written to. For optimal performance, the data pointed to SHOULD be 8-byte aligned. There is no endianness associated with the random data. On invocation the callback function will contain this
将被写入。为了获得最佳性能，指向的数据应该8字节对齐。没有与随机数据相关联的字符顺序。在调用时，回调函数将包含这个

parameter in its pOut parameter.

参数放在 pOut 参数中。

Return values:
返回值:

CPA_STATUS_SUCCESS Function executed successfully.
CPA_STATUS_SUCCESS 函数执行成功。
CPA_STATUS_FAIL Function failed.
CPA_STATUS_FAIL 函数失败。
CPA_STATUS_RETRY Resubmit the request.
CPA_STATUS_RETRY 重新提交请求。
CPA_STATUS_INVALID_PARAM Invalid parameter passed in.
传递的 CPA_STATUS_INVALID_PARAM 参数无效。
CPA_STATUS_RESOURCE Error related to system resources. One reason may be for an entropy test failing.
与系统资源相关的 CPA_STATUS_RESOURCE 错误。一个原因可能是
熵测试失败。
CPA_STATUS_RESTARTING API implementation is restarting. Resubmit the request.
CPA_STATUS_RESTARTING API 实现正在重新启动。重新提交请求。
CPA_STATUS_UNSUPPORTED Function is not supported.
不支持 CPA_STATUS_UNSUPPORTED 函数。

Precondition:
前提条件:

The component has been initialized via the **cpaCyStartInstance** function. The DRBG session, or instance, has been initialized via the **cpaCyDrbgInitSession** function.
该组件已通过 cpaCyStartInstance cpaCyDrbgInitSession

Postcondition:
后置条件:

None
没有人

CpaStatus cpaCyDrbgRemoveSession (const **CpaInstanceHandle** *instanceHandle*

CpaStatus cpaCyDrbgRemoveSession (const **CpaInstanceHandle**

Removes a previously instantiated DRBG session, or instance.
删除以前实例化的 DRBG 会话或实例。

This function will remove a previously initialized DRBG session, or instance, and the installed callback handler function. Removal will fail if outstanding calls still exist for the initialized session. In this case, the client needs to retry the remove function at a later time. The memory for the session handle MUST not be freed until this call has completed successfully.
此函数将删除先前初始化的 DRBG 会话或实例，以及已安装的回调处理函数。如果初始化的会话仍然存在未完成的调用，删除将会失败。在这种情况下，客户端需要稍后重试删除功能。在此调用成功完成之前，不能释放会话句柄的内存。

Context:
背景:

This is a synchronous function and it can sleep. It MUST NOT be executed in a context that DOES NOT permit sleeping.
这是一个同步功能，它可以休眠。它不能在不允许休眠的上下文中执行。

Assumptions:
假设:

None
没有人

Side-Effects:
副作用:

None
没有人

Blocking:
阻止:

No.
号码

Reentrant:
可重入:

No
不

Thread-safe:
线程安全:

Yes
是

Parameters:
参数:

`[in]` *instanceHandle* Instance handle.
[in] instanceHandle 执行个体控制代码。
`[in]` *sessionHandle* DRBG session handle to be removed.
[in] sessionHandle 要移除的 DRBG 会话句柄。

Return values:
返回值:

CPA_STATUS_SUCCESS Function executed successfully.
CPA_STATUS_SUCCESS 函数执行成功。
CPA_STATUS_FAIL Function failed.
CPA_STATUS_FAIL 函数失败。

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.

CPA_STATUS_RETRY 重新提交请求。传递的 CPA_STATUS_INVALID_PARAM 参数无效。与系统资源相关的 CPA_STATUS_RESOURCE 错误。CPA_STATUS_RESTARTING API 实现正在重新启动。重新提交请求。不支持 CPA_STATUS_UNSUPPORTED 函数。

Precondition:

前提条件:

The component has been initialized via the **cpaCyStartInstance** function. The DRBG session, or instance, has been initialized via the **cpaCyDrbgInitSession** function.

该组件已通过 **cpaCyStartInstance cpaCyDrbgInitSession**

Postcondition:

后置条件:

None

没有人

CpaStatus cpaCyDrbgQueryStats64 (const **CpaInstanceHandle** *instanceHandle*,
CpaCyDrbgStats64 * *pStats*

CpaStatus cpaCyDrbgQueryStats64 (const **CpaInstanceHandle** *instanceHandle*,
CpaCyDrbgStats64 * pStats

Returns statistics specific to a session, or instance, of the RBG API.

返回特定于 RBG API 会话或实例的统计信息。

This function will query a specific session for RBG statistics. The user MUST allocate the CpaCyDrbgStats64 structure and pass the reference to that into this function call. This function writes the statistic results into the passed in CpaCyDrbgStats64 structure.

该函数将查询特定会话的 RBG 统计数据。用户必须分配 CpaCyDrbgStats64 结构，并将对该结构的引用传递到此函数调用中。此函数将统计结果写入传入的 CpaCyDrbgStats64 结构中。

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.

注意:此函数返回的统计数据不会中断当前的数据处理，因此可能会与统计数据检索过程中正在进行的操作稍微不同步。

Context:

背景:

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

这是一个同步功能，它可以休眠。它不能在不允许休眠的上下文中执行。

Assumptions:

假设:

None

没有人

Side-Effects:

副作用:

None

没有人

Blocking:
阻止:

This function is synchronous and blocking.
这个函数是同步的和阻塞的。

Reentrant:
可重入:

No
不

Thread-safe:
线程安全:

Yes
是

Parameters:
参数:

`[in]` *instanceHandle* Instance handle.
[in] instanceHandle 执行个体控制代码。
`[out]` *pStats* Pointer to memory into which the statistics will be written.
[out] pStats 指向将写入统计信息的内存的指针。

Return values:
返回值:

CPA_STATUS_SUCCESS Function executed successfully.
CPA_STATUS_SUCCESS 函数执行成功。
CPA_STATUS_FAIL Function failed.
CPA_STATUS_INVALID_PARAM Invalid parameter passed in.
CPA_STATUS_RESOURCE Error related to system resources.
CPA_STATUS_FAIL 函数失败。传递的 CPA_STATUS_INVALID_PARAM 参数无效。与系统资源相关的 CPA_STATUS_RESOURCE 错误。
CPA_STATUS_RESTARTING API implementation is restarting. Resubmit the request.
CPA_STATUS_RESTARTING API 实现正在重新启动。重新提交请求。
CPA_STATUS_UNSUPPORTED Function is not supported.
不支持 CPA_STATUS_UNSUPPORTED 函数。

Precondition:

前提条件:

Component has been initialized.

组件已初始化。

Postcondition:

后置条件:

None

没有人

26 Non-Deterministic Random Bit Generation API

27 非确定性随机位生成API

[Cryptographic API]

[Cryptographic API]

Collaboration diagram for Non-Deterministic Random Bit Generation API:
非确定性随机位生成API的协作图:

Cryptographic API
Non-Deterministic Random Bit Generation API
Cryptographic API
Non-Deterministic Random Bit Generation API

27.1 Detailed Description

27.2 详细描述

File: cpa_cy_nrbg.h

文件:cpa_cy_nrbg.h

These functions specify the API for Non-Deterministic Random Bit Generation (NRBG). This is used to provide entropy to a Deterministic RBG (DRBG).
这些函数指定了用于非确定性随机位生成(NRBG)的API。这用于向确定性RBG (DRBG)提供熵。

Note:
注意:

> These functions supersede the random number generation functions in API group **Random Bit/Number Generation API**, which are now deprecated.
> 这些函数取代了API组中的随机数生成函数**RandomBit/Number Generation API**

27.3 Data Structures

27.4 数据结构

- struct **_CpaCyNrbgOpData**
- 结构体_CpaCyNrbgOpData

27.5 Typedefs

27.6 类型定义

- typedef **_CpaCyNrbgOpData CpaCyNrbgOpData**
- 数据类型说明_CpaCyNrbgOpData CpaCyNrbgOpData

27.7 Functions

27.8 功能

- **CpaStatus cpaCyNrbgGetEntropy** (const **CpaInstanceHandle** instanceHandle, const **CpaCyGenFlatBufCbFunc** pCb, void *pCallbackTag, const **CpaCyNrbgOpData** *pOpData, **CpaFlatBuffer** *pEntropy)
- CpaStatus cpaCyNrbgGetEntropy（常量 CpaInstanceHandle CpaCyGenFlatBufCbFunc CpaCyNrbgOpData CpaFlatBuffer

27.9 Data Structure Documentation

27.10 数据结构文档

20.5.1 _CpaCyNrbgOpData Struct Reference

20.5.2 _CpaCyNrbgOpData 结构引用

20.5.2.1 Detailed Description

20.5.2.2 详细描述

NRBG Get Entropy Operation Data
NRBG 获取熵运算数据

This structure contains data relating to generation of entropy using an NRBG.
该结构包含与使用 NRBG 产生熵相关的数据。

See also:
另请参见:

cpaCyNrbgGetEntropy()
cpaCyNrbgGetEntropy()

Note:

注意:

义

If the client modifies or frees the memory referenced in this structure after it has been submitted to the 如果客户端在将此结构提交给 **cpaCyNrbgGetEntropy()** function, and before it has been returned in the callback, undefined cpaCyNrbgGetEntropy() 函数，并且在回调中返回之前，未定

20.5.1 _CpaCyNrbgOpData Struct Reference

behavior will result.

20.5.1 _CpaCyNrbgOpData 结构引用行为将导

致。

20.5.2.3 Data Fields

20.5.2.4 数据字段

- **Cpa32U lengthInBytes**
- Cpa32U lengthInBytes

20.5.2.5 Field Documentation

20.5.2.6 现场文件

Cpa32U _CpaCyNrbgOpData::lengthInBytes

Requested number of bytes to be generated. On calls to **cpaCyNrbgGetEntropy**, this value must be greater than zero (>0).

请求生成的字节数。打电话给 cpaCyNrbgGetEntropy

27.11 Typedef Documentation

27.12 Typedef 文档

typedef struct _**CpaCyNrbgOpData CpaCyNrbgOpData**

NRBG Get Entropy Operation Data

NRBG 获取熵运算数据

This structure contains data relating to generation of entropy using an NRBG.

该结构包含与使用 NRBG 产生熵相关的数据。

See also:

另请参见：

cpaCyNrbgGetEntropy()

cpaCyNrbgGetEntropy()

Note:

注意：

If the client modifies or frees the memory referenced in this structure after it has been submitted to the **cpaCyNrbgGetEntropy()** function, and before it has been returned in the callback, undefined behavior will result.
如果客户端在将此结构提交给 cpaCyNrbgGetEntropy()

27.13 Function Documentation

27.14 功能文档

cpaCyNrbgGetEntropy (const instanceHandle,

```
cpaCyNrbgGetEntropy  (  const                                        instanceHandle,
                             const   pCb,
                             void *pCallbackTag,
                             const *pOpData,
                             *pEntropy
                            )
```

Gets entropy from the NRBG.
从 NRBG 获取熵。

This function returns a string of bits of specified length.
这个函数返回一串指定长度的位。

Context:
背景:

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.
当作为异步函数调用时，它不能休眠。它可以在不允许休眠的上下文中执行。当作为同步函数调用时，它可能会休眠。它不能在不允许休眠的上下文中执行。

Assumptions:
假设:

None
没有人

Side-Effects:
副作用:

None
没有人

Blocking:
阻止:

Yes when configured to operate in synchronous mode.

当配置为在同步模式下运行时，是。

Reentrant:
可重入:

No
不

Thread-safe:
线程安全:

Yes
是

Parameters:
参数:

`[in]` *instanceHandle* Instance handle.
[in] instanceHandle 执行个体控制代码。
`[in]` *pCb* Pointer to callback function to be invoked when the operation is complete.
[in]作业完成时要叫用的回呼函式的 pCb 指标。
If this is set to a NULL value the function will operate synchronously.
如果设置为空值，函数将同步运行。
`[in]` *pCallbackTag* Opaque User Data for this specific call. Will be returned unchanged in the callback.
[in] pCallbackTag 此特定调用的不透明用户数据。将在回调中不变地返回。
`[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.
[in] pOpData 结构，包含执行作业所需的所有资料。客户端代码为此结构分配内存。该组件取得内存的所有权，直到它在回调中被返回。
`[out]` *pEntropy* Pointer to memory allocated by the client to which the entropy will be written. For optimal performance, the data pointed to SHOULD be 8-byte aligned. There is no endianness associated with the entropy. On invocation the callback function will contain this parameter in its pOut parameter.
[out]pen ropy 指标，指向将写入熵的用户端所配置的记忆体。为了获得最佳性能，指向的数据应该 8 字节对齐。熵与字节序无关。在调用时，回调函数将在 pOut 参数中包含这个参数。

Return values:

返回值:

CPA_STATUS_SUCCESS Function executed successfully.
CPA_STATUS_SUCCESS 函数执行成功。
CPA_STATUS_FAIL Function failed.
CPA_STATUS_FAIL 函数失败。
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.
CPA_STATUS_RETRY 重新提交请求。传递的 CPA_STATUS_INVALID_PARAM 参数无效。与系统资

源相关的 CPA_STATUS_RESOURCE 错误。CPA_STATUS_RESTARTING API 实现正在重新启动。重新提交请求。不支持 CPA_STATUS_UNSUPPORTED 函数。

Precondition:

前提条件:

The component has been initialized via the **cpaCyStartInstance** function.

该组件已通过 cpaCyStartInstance

Postcondition:

后置条件:

None

没有人

Note:

注意:

为非空时，类型的异步回调 CpaCyGenFlatBufCbFunc

When pCb is non-NULL an asynchronous callback of type **CpaCyGenFlatBufCbFunc** 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.

当 pCb

28 Random Bit/Number Generation API

29 随机位/数生成API

[Cryptographic API]

[Cryptographic API]

Collaboration diagram for Random Bit/Number Generation API:
随机位/数生成API的协作图:

Cryptographic API
Random Bit/Number Generation API
Cryptographic API
Random Bit/Number Generation API

29.1 Detailed Description

29.2 详细描述

File: cpa_cy_rand.h

Deprecated:

文

件:cpa_cy_rand.hDe

precated:

> As of v1.3 of the API, this entire API group has been deprecated, replaced by API groups
> 从API的v1.3开始，整个API组已经被废弃，由API组代替
> **Deterministic Random Bit Generation API** and **Non-Deterministic Random Bit Generation API** .

These functions specify the API for the Cryptographic Random Bit and Random number generation.

> Deterministic Random Bit Generation API 和Non-Deterministic Random Bit Generation API

29.3 Data Structures

29.4 数据结构

- struct **_CpaCyRandStats**

- 结构体_CpaCyRandStats
- struct **_CpaCyRandGenOpData**
- 结构体_CpaCyRandGenOpData
- struct **_CpaCyRandSeedOpData**
- 结构体_CpaCyRandSeedOpData

29.5 Defines

29.6 界定

- #define **CPA_CY_RAND_SEED_LEN_IN_BYTES**

- #定义 CPA_CY_RAND_SEED_LEN_IN_BYTES

29.7 Typedefs

29.8 类型定义

- typedef **_CpaCyRandStats CPA_DEPRECATED**

- 数据类型说明_CpaCyRandStats CPA_DEPRECATED
- typedef **_CpaCyRandGenOpData CPA_DEPRECATED**
- 数据类型说明_CpaCyRandGenOpData CPA_DEPRECATED
- typedef **_CpaCyRandSeedOpData CPA_DEPRECATED**
- 数据类型说明_CpaCyRandSeedOpData CPA_DEPRECATED

29.9 Functions

29.10 功能

- **CpaStatus CPA_DEPRECATED cpaCyRandGen** (const **CpaInstanceHandle** instanceHandle, const **CpaCyGenFlatBufCbFunc** pRandGenCb, void *pCallbackTag, const struct

- CpaStatus CPA_DEPRECATED cpaCyRandGen（常量CpaInstanceHandle CpaCyGenFlatBufCbFunc
  _CpaCyRandGenOpData *pRandGenOpData, **CpaFlatBuffer** *pRandData)
  _CpaCyRandGenOpData *pRandGenOpData，CpaFlatBuffer
- **CpaStatus CPA_DEPRECATED cpaCyRandSeed** (const **CpaInstanceHandle** instanceHandle, const **CpaCyGenericCbFunc** pRandSeedCb, void *pCallbackTag, const struct
- CpaStatus CPA_DEPRECATED cpaCyRandSeed（常量CpaInstanceHandle CpaCyGenericCbFunc
  _CpaCyRandSeedOpData *pSeedOpData)
  _CpaCyRandSeedOpData *pSeedOpData)
- **CpaStatus CPA_DEPRECATED cpaCyRandQueryStats** (const **CpaInstanceHandle**
- CpaStatus CPA_DEPRECATED cpaCyRandQueryStats（常量CpaInstanceHandle
  instanceHandle, struct **_CpaCyRandStats** *pRandStats)
  instanceHandle，结构_CpaCyRandStats

29.11 Data Structure Documentation

29.12 数据结构文档

21.6.1 _CpaCyRandStats Struct Reference

21.6.1 _CpaCyRandStats 结构引用

21.6.1 _CpaCyRandStats Struct Reference

21.6.2 _CpaCyRandStats 结构引用

21.6.2.1 Detailed Description

Random Data Generator Statistics.

Deprecated:

21.6.2.2 随机数据生成器统

计。Deprecated:

As of v1.3 of the API, replaced by **CpaCyDrbgStats64**.
自 API 1.3 版起，由以下内容取代 CpaCyDrbgStats64

This structure contains statistics on the random data generation operations. Statistics are set to zero when the component is initialized, and are collected per instance.
此结构包含随机数据生成操作的统计信息。当组件初始化时，统计信息被设置为零，并针对每个实例进行收集。

21.6.2.3 Data Fields
21.6.2.4 数据字段

- **Cpa32U numRandNumRequests**
- Cpa32U numRandNumRequests
- **Cpa32U numRandNumRequestErrors**
- Cpa32U numRandNumRequestErrors
- **Cpa32U numRandNumCompleted**
- Cpa32U numRandNumCompleted
- **Cpa32U numRandNumCompletedErrors**
- Cpa32U numRandNumCompletedErrors
- **Cpa32U numRandBitRequests**
- Cpa32U numRandBitRequests
- **Cpa32U numRandBitRequestErrors**
- Cpa32U numRandBitRequestErrors
- **Cpa32U numRandBitCompleted**
- Cpa32U numRandBitCompleted
- **Cpa32U numRandBitCompletedErrors**
- Cpa32U numRandBitCompletedErrors
- **Cpa32U numNumSeedRequests**
- Cpa32U numNumSeedRequests
- **Cpa32U numRandSeedCompleted**
- Cpa32U numRandSeedCompleted
- **Cpa32U numNumSeedErrors**
- Cpa32U numNumSeedErrors

21.6.2.5 Field Documentation
21.6.2.6 现场文件

Cpa32U CpaCyRandStats::numRandNumRequests

Total number of successful random number generation requests.
成功的随机数生成请求的总数。

Cpa32U CpaCyRandStats::numRandNumRequestErrors

Total number of random number generation requests that had an error and could not be processed.
出错且无法处理的随机数生成请求的总数。

Cpa32U CpaCyRandStats::numRandNumCompleted

Total number of random number operations that completed successfully.
成功完成的随机数操作的总数。

Cpa32U CpaCyRandStats::numRandNumCompletedErrors

Total number of random number operations that could not be completed successfully due to errors.
由于错误而无法成功完成的随机数操作的总数。

Cpa32U CpaCyRandStats::numRandBitRequests

Total number of successful random bit generation requests.
成功的随机位生成请求的总数。

Cpa32U CpaCyRandStats::numRandBitRequestErrors

Total number of random bit generation requests that had an error and could not be processed.
有错误且无法处理的随机位生成请求的总数。

Cpa32U CpaCyRandStats::numRandBitCompleted

Total number of random bit operations that completed successfully.
成功完成的随机位操作的总数。

Cpa32U CpaCyRandStats::numRandBitCompletedErrors

Total number of random bit operations that could not be completed successfully due to errors.
由于错误而无法成功完成的随机位操作的总数。

Cpa32U _CpaCyRandStats::numNumSeedRequests

Total number of seed operations requests.
种子操作请求的总数。

Cpa32U _CpaCyRandStats::numRandSeedCompleted

Total number of seed operations completed.
已完成的种子操作总数。

Cpa32U _CpaCyRandStats::numNumSeedErrors

Total number of seed operation errors.
种子操作错误的总数。

21.6.2 _CpaCyRandGenOpData Struct Reference

21.6.3 _CpaCyRandGenOpData 结构引用

21.6.3.1 Detailed Description

Random Bit/Number Generation Data.

Deprecated:

21.6.3.2 详细描述随机位/数生成数

据。Deprecated:

As of v1.3 of the API, replaced by **CpaCyDrbgGenOpData**.
自 API 1.3 版起，由以下内容取代 CpaCyDrbgGenOpData

This structure lists the different items that are required in the cpaCyRandGen 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.
此结构列出了 cpaCyRandGen 函数中所需的不同项目。客户端必须为这个结构分配内存。当结构被传递给函数时，内存的所有权就传递给了函数。当这个结构随回调一起返回时，内存的所有权返回给客户端。

Note:
注意:

If the client modifies or frees the memory referenced in this structure after it has been submitted to the cpaCyRandGen function, and before it has been returned in the callback, undefined behavior will result.
如果客户端在将此结构中引用的内存提交给 cpaCyRandGen 函数之后，在回调中返回之前修改或释放该内存，将导致未定义的行为。

21.6.3.3 Data Fields
21.6.3.4 数据字段

- **CpaBoolean generateBits**
- CpaBoolean generateBits
- **Cpa32U lenInBytes**
- Cpa32U lenInBytes

21.6.3.5 Field Documentation
21.6.3.6 现场文件

CpaBoolean CpaCyRandGenOpData::generateBits

When set to CPA_TRUE then the cpaCyRandGen function will generate random bits which will comply with the ANSI X9.82 Part 1 specification. When set to CPA_FALSE random numbers will be produced from the random bits generated by the hardware. This will be spec compliant in terms of the probability of the random nature of the number returned.

当设置为CPA_TRUE时，cpaCyRandGen函数将产生符合ANSI X9.82第1部分规范的随机位。当设置为CPA_FALSE时，将从硬件产生的随机位中产生随机数。就返回数字的随机性质的概率而言，这将是符合规范的。

Cpa32U CpaCyRandGenOpData::lenInBytes

Specifies the length in bytes of the data returned. If the data returned is a random number, then it is implicit that the random number will fall into the following range: Expressed mathematically, the range is [2^(lenInBytes*8 - 1) to 2^(lenInBytes*8) - 1]. This is equivalent to "1000...0000" to "1111...1111" which requires (lenInBytes * 8) bits to represent. The maximum number of random bytes that can be requested is 65535 bytes.

指定返回数据的字节长度。如果返回的数据是一个随机数，那么就隐含着这个随机数将落入以下范围:用数学的方式表示，这个范围是[2^(leninbytes*8-1]到2^(lenInBytes*8) - 1]。这相当于“1000...0000 ″到″1111...这需要(lenInBytes * 8)个比特来表示。可以请求的最大随机字节数是65535字节。