# Intel® QuickAssist Technology Cryptographic API Reference

# 英特尔快速辅助技术加密 API 参考

***Automatically generated from sources, July 14, 2021.***
*根据来源自动生成，2021 年 7 月 14 日。*

***Based on API version 2.5***

*基于 API 2.5 版*

***(See Release Notes to map API version to software package version.)***

*（参见发行说明，将 API 版本映射到软件包版本。）*

**Reference Number: 330685-007**
参考编号:330685-007

By using this document, in addition to any agreements you have with Intel, you accept the terms set forth below. You may not use or facilitate the use of this document in connection with any infringement or other legal analysis concerning Intel products described herein. You agree to grant Intel a non-exclusive, royalty-free license to any patent claim thereafter drafted which includes subject matter disclosed herein.

通过使用本文档，除了您与英特尔达成的任何协议之外，您还接受以下条款。您不得将本文档用于或协助用于与本文所述英特尔产品相关的任何侵权或其他法律分析。您同意授予英特尔一项非独家、免版税的许可，允许英特尔享有此后起草的任何专利声明，包括此处披露的主题。

INFORMATION IN THIS DOCUMENT IS PROVIDED IN CONNECTION WITH INTEL PRODUCTS. NO LICENSE, EXPRESS OR IMPLIED, BY ESTOPPEL OR OTHERWISE, TO ANY INTELLECTUAL PROPERTY RIGHTS IS GRANTED BY THIS DOCUMENT. EXCEPT AS PROVIDED IN INTEL'S TERMS AND CONDITIONS OF SALE FOR SUCH PRODUCTS, INTEL ASSUMES NO LIABILITY WHATSOEVER AND INTEL DISCLAIMS ANY EXPRESS OR IMPLIED WARRANTY, RELATING TO SALE AND/OR USE OF INTEL PRODUCTS INCLUDING LIABILITY OR WARRANTIES RELATING TO FITNESS FOR A PARTICULAR PURPOSE, MERCHANTABILITY, OR INFRINGEMENT OF ANY PATENT, COPYRIGHT OR OTHER INTELLECTUAL PROPERTY RIGHT.

本文档中的信息与英特尔产品相关。本文档未通过禁止反言或其他方式授予任何知识产权的许可，无论是明示的还是暗示的。除了此类产品的英特尔销售条款和条件中规定的以外，英特尔不承担任何责任，并且英特尔否认与销售和 /或使用英特尔产品相关的任何明示或暗示的担保，包括与特定用途适用性、适销性或不侵犯任何专利、版权或其他知识产权相关的责任或担保。

A "Mission Critical Application" is any application in which failure of the Intel Product could result, directly or indirectly, in personal injury or death. SHOULD YOU PURCHASE OR USE INTEL'S PRODUCTS FOR ANY SUCH MISSION CRITICAL APPLICATION, YOU SHALL INDEMNIFY AND HOLD INTEL AND ITS SUBSIDIARIES, SUBCONTRACTORS AND AFFILIATES, AND THE DIRECTORS, OFFICERS, AND EMPLOYEES OF EACH, HARMLESS AGAINST ALL CLAIMS COSTS, DAMAGES, AND EXPENSES AND REASONABLE ATTORNEYS' FEES ARISING OUT OF, DIRECTLY OR INDIRECTLY, ANY CLAIM OF PRODUCT LIABILITY, PERSONAL INJURY, OR DEATH ARISING IN ANY WAY OUT OF SUCH MISSION CRITICAL APPLICATION, WHETHER OR NOT INTEL OR ITS SUBCONTRACTOR WAS NEGLIGENT IN THE DESIGN, MANUFACTURE, OR WARNING OF THE INTEL PRODUCT OR ANY OF ITS PARTS.

“关键任务应用”是指英特尔产品的故障可能直接或间接导致人身伤害或死亡的任何应用。如果您为任何此类关键任务应用程序购买或使用英特尔产品，您应赔偿并保护英特尔及其子公司、分包商和附属公司，以及各自的董事、高级职员和员工，使其免于承担因直接或间接的任何此类关键任务应用程序引起的任何产品责任、人身伤害或死亡索赔而产生的所有索赔费用、损害、开支和合理的律师费，无论英特尔或其分包商是否在英特尔产品或其任何部分的设计、制造或警告方面存在疏忽。

Intel may make changes to specifications and product descriptions at any time, without notice. Designers must not rely on the absence or characteristics of any features or instructions marked "reserved" or "undefined". Intel reserves these for future definition and shall have no responsibility whatsoever for conflicts or incompatibilities arising from future changes to them. The information here is subject to change without notice. Do not finalize a design with this information.

英特尔可能随时更改规格和产品说明，恕不另行通知。设计者不得依赖任何标有“保留”或“未定义”的特征或说明的缺失或特征。英特尔保留这些内容以供将来定义，对于未来对它们的更改所引起的冲突或不兼容性，英特尔不承担任何责任。此处的信息如有更改，恕不另行通知。不要用这些信息来完成设计。

The products described in this document may contain design defects or errors known as errata which may cause the product to deviate from published specifications. Current characterized errata are available on request.

本文档中描述的产品可能包含勘误表中已知的设计缺陷或错误，这可能会导致产品偏离公布的规格。现有的特征勘误表可应要求提供。

Contact your local Intel sales office or your distributor to obtain the latest specifications and before placing your product order.

在下产品订单之前，请联系您当地的英特尔销售办事处或经销商以获取最新的规格。

Copies of documents which have an order number and are referenced in this document, or other Intel literature, may be obtained by calling 1-800-548-4725, or go to: http://www.intel.com/design/literature.htm.

可致电 1-800-548-4725 获取本文档或其他英特尔文献中引用的有订单号的文档副本，或访问 :http://www.intel.com/design/literature.htm.

Any software source code reprinted in this document is furnished for informational purposes only and may only be used or copied and no license, express or implied, by estoppel or otherwise, to any of the reprinted source code is granted by this document.

本文档中重印的任何软件源代码仅供参考，只能使用或复制，本文档并未通过禁止反言或其他方式授予任何重印源代码任何明示或暗示的许可。

This document contains information on products in the design phase of development.

本文件包含开发设计阶段的产品信息。

Intel processor numbers are not a measure of performance. Processor numbers differentiate features within each processor family, not across different processor families. Go to: http://www.intel.com/products/processor_number/.
英特尔处理器号不是性能的衡量标准。处理器编号区分每个处理器家族内的特性，而不是不同处理器家族之间的特性。转到:http://www.intel.com/products/processor_number/.

Code Names are only for use by Intel to identify products, platforms, programs, services, etc. (â“ productsâ“ ) in development by Intel that have not been made commercially available to the public, i.e., announced, launched or shipped. They are never to be used as â“ commercialâ“ names for products. Also, they are not intended to function as trademarks.
代码名称仅供英特尔用于识别产品、平台、计划、服务等。由英特尔开发的尚未向公众公开上市(即宣布、推出或发货)的(产品)。它们决不能用作产品的“商业”名称。此外，它们也无意用作商标。

Intel and the Intel logo are trademarks of Intel Corporation in the U.S. and/or other countries.
英特尔和英特尔标志是英特尔公司在美国和/或其他国家的商标。



Reference Number: 330685-007
参考编号:330685-007

# Revision History
# 修订历史

| Date | Revision | Description |
|---|---|---|
| May 2021 | 008 | Changed version of the crypto API to v2.5<br><br>Added support for SM2. |
| November 2020 | 007 | Changed version of the crypto API to v2.4<br><br>Added support for ChaCha20-Poly1305.<br><br>Added support for SM4 in ECB, CBC and CTR modes.<br><br>Added support for SM3.<br><br>Added support for SHA3-224, SHA3-384 and SHA3-512. |
| March 2020 | 006 | Added HKDF API.<br><br>Added 25519 and 448 curve support to cpa_cy_ec.h. |
| April 2018 | 005 | Added session update API. |
| July 2016 | 004 | Added Intel® Key Protection Technology (KPT) API. |
| October 2015 | 003 | Changed version of the crypto API to v2.0.<br><br>Added ZUC-EEA3 and ZUC-EIA3 support to the crypto API.<br><br>Added SHA3-256 support to the crypto API. |
| September 2015 | 002 | Incrementing CY API version number to v1.9.<br><br>Adding CPA_STATUS_UNSUPPORTED as a return status for each function and callback. |
| June 2014 | 001 | First “public” version of the document. Based on “Intel Confidential” document number 410923-1.8 with the revision history of that document retained for reference purposes. |
| April 2014 | 1.8 | Resolves the following work requests:<br><br>· Fixing specification of __FreeBSD<br><br>· Whitespace clean-up<br><br>· IXA00384099: Adding default 'None' entries to CpaCySymOp and CpaCySymHashAlgorithm<br><br>· IXA00384099: Addition of CPA_CY_SYM_HASH_AES_CBC_MAC<br><br>· IXA00384492: Addition of cpaCySymSessionCtxGetDynamicSize() and cpaCySymDpSessionCtxGetDynamicSize()<br><br>· IXA00385073: Added performance guidance notes for source buffer lengths on the crypto API. |
| February 2013 | 1.7 | Addition of AES-XTS mode |
| November 2012 | 1.6-RC2 | Resolves the following work requests:<br><br>· TECG00000192: Complete AES-GMAC support |

| 日期 | 修订本 | 描述 |
| --- | --- | --- |
| 2021 年 5 月 | 008 | 将加密 API 的版本更改为 2.5 版<br><br>增加了对 SM2 的支持。 |
| 2020 年 11 月 | 詹姆斯·邦德 | 将加密 API 的版本更改为 2.4 版，增加了<br><br>对 ChaCha20-Poly1305 的支持。<br><br>增加了对 ECB、CBC 和 CTR 模式下 SM4 的支持。增加了对<br><br>SM3 的支持。<br><br>增加了对 SHA3-224、SHA3-384 和 SHA3-512 的支持。 |
| 2020 年 3 月 | 006 | 增加了 HKDF API。<br><br>cpa_cy_ec.h 增加了 25519 和 448 曲线支持。 |
| 2018 年 4 月 | 005 | 添加了会话更新 API。 |
| 2016 年 7 月 | 004 | 添加了英特尔密钥保护技术(KPT) API。 |
| 2015 年十月 | 003 | 已将加密 API 的版本更改为 2.0 版<br><br>为加密 API 添加了 ZUC-EEA3 和 ZUC-EIA3 支持。向加密 API<br><br>添加了 SHA3-256 支持。 |
| 2015 年 9 月 | 002 | 将 CY API 版本号增加到 v1.9。<br><br>添加 CPA_STATUS_UNSUPPORTED 作为每个函数和回调的返回状态。 |
| 2014 年 6 月 | 001 | 文档的第一个“公共”版本。基于编号为 410923-1.8 的“英特尔机密”文档，保留该文档的修订历史以供参考。 |
| 2014 年 4 月 | 1.8 | 解决以下工作请求:<br><br>· FreeBSD 的安装规范<br><br>· 空白清除<br><br>· IXA00384099:向 CpaCySymOp 和 CpaCySymHashAlgorithm 添加默认的“None”条目<br><br>· IXA00384099:添加 CPA _ CY _ SYM _哈希_AES_CBC_MAC<br><br>· IXA00384492:添加了 cpacysymsessiontxgetdynamicsize()和 cpacysymdpsessiontxgetdynamicsize()<br><br>· IXA00385073:为 crypto API 上的源缓冲区长度添加了性能指导说明。 |
| 2013 年 2 月 | 1.7 | 增加了 AES-XTS 模式 |
| 2012 年 11 月 | 1.6-RC2 | 解决以下工作请求:<br><br>TECG00000192:完全支持 AES-GMAC |

| | | |
|---|---|---|
| October 2012 | 1.5 | Resolves the following work requests:<br><br>· TECG00000186: Add instance notification support for RESTARTING & RESTARTED events and CPA_STATUS_RESTARTING return codes. |
| October 2012 | 1.6-RC1 | Resolves the following work requests:<br><br>· TECG00000187: Add support for AES-F8<br><br>· TECG00000189: Add a unique instance identifier to CpaInstanceInfo2 |
| June 2012 | 1.4 | Resolves comments against previous revision.<br><br>Resolves the following work requests:<br><br>· TECG00000178: Removing CPA_CY_KEY_GEN_SSL_TLS_SEED_LEN_IN_BYTES from cpa_cy_key.h<br><br>· TECG00000180: Adding detail on GMAC to API comments.<br><br>· TECG00000181: Update RSA comments to call out no padding.<br><br>· TECG00000182: DSA FIPS PUB 186-2 with Change Notice 1 updates to supported DSA key lengths.<br><br>· TECG00000183: Clarifying that the message buffers may not be cleared when using the DP API if digest verification fails for CCM/GCM. |
| May 2012 | 1.3 | Resolves the following work requests:<br><br>· TECG00000175: Add support partial packets for chained operations and nested hash operations.<br><br>· TECG00000162: Removed references to digestVerify and updated description of pDigestResult.<br><br>· TECG00000167: cpaCyDhKeyGenPhase1 does not generate private value (x) on CCK |
| Apr 2012 | 1.3-RC15 | Resolves the following work requests:<br><br>· TECG00000169: Removing CPA_CY_SYM_DP_TMP_WORKAROUND from cpa_cy_sym_dp.h<br><br>· TECG00000170: (IXA00372445 ) Updated API comments to say that it is safe to assume that cpaCySymDpSessionCtxGetSize() will always return the same size for a given implementation. Same for cpaCySymSessionCtxGetSize(). |
| Mar 2012 | 1.3-RC14 | Resolves the following work requests:<br><br>· TECG00000166: Added ability to query bus address information for a CpaInstance. |
| Nov 2011 | 1.3-RC13 | Resolved comments against RC12. |
| Oct 2011 | 1.3-RC12 | Resolves the following work requests: |

| | | |
|---|---|---|
| 2012 年 10 月 | 1.5 | 解决以下工作请求:<br><br>TECG00000186:为重新启动& RESTARTED 事件和 CPA _ STATUS _ RESTARTING 返回代码添加实例通知支持。 |
| 2012 年 10 月 | 1.6-RC1 | 解决以下工作请求:<br><br>· TECG00000187:添加对 AES-F8 的支持<br><br>· TECG00000189:向 CpaInstanceInfo2 添加唯一的实例标识符 |
| 2012 年 6 月 | 1.4 | 根据以前的修订解决注释。解决以下工作请求:<br><br>· TECG00000178:从中删除 CPA _ CY _ KEY _ GEN _ SSL _ TLS _ SEED _ LEN _ IN _ BYTES cpa_cy_key.h<br><br>· TECG00000180:将 GMAC 的详细信息添加到 API 注释中。<br><br>· TECG00000181:更新 RSA 注释以调用无填充。<br><br>· TECG00000182: DSA FIPS 发布 186-2，变更通知 1 更新了支持的 DSA 密钥长度。<br><br>· TECG00000183:澄清了当使用 DP API 时，如果 CCM/GCM 的摘要验证失败，消息缓冲区可能不会被清除。 |
| 2012 年 5 月 | 1.3 | 解决以下工作请求:<br><br>· TECG00000175:添加对链式操作和嵌套哈希操作的部分数据包的支持。<br><br>· TECG00000162:删除了对 digestVerify 的引用并更新了 pDigestResult 的描述。<br><br>· tecg 00000167:cpacydhkeygenphase 1 不在 CCK 上生成私有值(x) |
| 2012 年 4 月 | 1.3-RC15 | 解决以下工作请求:<br><br>· TECG00000169:从中删除 CPA _ CY _ SYM _ DP _ TMP _变通办法 cpa_cy_sym_dp.h<br><br>· TECG00000170: (IXA00372445)更新了 API 注释，说明可以安全地假设对于给定的实现，cpaCySymDpSessionCtxGetSize()将始终返回相同的大小。对于 cpaCySymSessionCtxGetSize()也是如此。 |
| 2012 年 3 月 | 1.3-RC14 | 解决以下工作请求:<br><br>TECG00000166:增加了为 CpaInstance 查询总线地址信息的能力。 |
| 2011 年 11 月 | 1.3-RC13 | 针对 RC12 的已解决意见。 |

| | | |
|---|---|---|
| 2011 年 10 月 | 1.3-RC12 | 解决以下工作请求:<br><br>· TECG00000135: Updated comments on key generation API with references to RFC5246 (TLS v1.2)<br><br>· TECG00000147: Added hashAlgorithm parameter to TLS v1.2 PRF function<br><br>· TECG00000153: Clarified cases when digest result should point to src vs. dst buffer<br><br>· TECG00000154: Documented that verification failure for GCM/CCM will not result in the buffer being zeroised. Also added flag on DP API to indicate whether digestIsEncrypted.<br><br>· TECG00000155: Removed parameter (number of requests submitted) from "perform op now" function<br><br>· TECG00000156: Documented that some "unused" fields are in fact reserved for internal usage. |
| Jul 2011 | 1.3-RC11 | Updated DP API per feedback from engineering during implementation |
| Jun 2011 | 1-3-RC10 | Resolves comments against previous revisions, including the “traditional” and data plane APIs.<br><br>Also includes updates for the following work requests:<br><br>· TECG00000119: clarified max length for aadLenInBytes<br><br>· TECG00000120: added support for 512-bit RSA operations<br><br>· TECG00000121: added support for TLS 1.2 PRF/key generation function<br><br>· TECG00000082: added support for batch submission of requests (via data plane API)<br><br>· TECG00000030: clarified how large numbers are represented on the API |
| Apr 2011 | 1.3-RC8 | Adds the data plane API for symmetric crypto, specifically file cpa_cy_sym_dp.h. Also adds new types to represent flag buffers and buffer lists with physical addressing. |

| | | |
|---|---|---|
| Apr 2011 | 1.3-RC9 | Resolves the following issues/work requests:<br><br>· TECG00000098: drbg: Clarified description of reseed counter.<br><br>· TECG00000108: keygen: Updated description of MGF function to refer to PKCS#1 MGF1 function. Also added @ref to some Doxygen comments to prettify the documentation.<br><br>· TECG00000101: nrbg: Clarified that length of requested entropy must be >0<br><br>· TECG00000097: prime: updated the list of bit-sizes of prime number candidates supported<br><br>· TECG00000117: Updated description of various fields for GCM and CCM, specifically to allow these algorithms to be implemented entirely underneath the API and therefore enabling the implementations to be FIPS certified under CAVP<br><br>· TECG00000135:更新了关于引用 RFC5246 (TLS v1.2)的密钥生成 API 的注释<br><br>· TECG00000147:向 TLS v1.2 PRF 函数添加了 hashAlgorithm 参数<br><br>· TECG00000153:澄清了摘要结果应指向 src 与 dst 缓冲区的情况<br><br>· TECG00000154:记录 GCM/CCM 验证失败不会导致缓冲区归零。还在 DP API 上添加了标志来指示 digestIsEncrypted。<br><br>· TECG00000155:从“立即执行 op”函数中删除了参数(提交的请求数)<br><br>· TECG00000156:记录了一些“未使用的”字段实际上是为内部使用而保留的。 |
| 2011 年 7 月 | 1.3-RC11 | 根据工程实施期间的反馈更新 DP API |
| 2011 年 6 月 | 1-3-RC10 | 解决针对以前版本的注释，包括“传统”和数据平面 API。<br><br>还包括以下工作请求的更新:<br><br>· TECG00000119:阐明了 aadLenInBytes 的最大长度<br><br>· TECG00000120:增加了对 512 位 RSA 运算的支持<br><br>· TECG00000121:增加了对 TLS 1.2 PRF/密钥生成功能的支持<br><br>· TECG00000082:增加了对批量提交请求的支持(通过数据平面 API)<br><br>· TECG00000030:阐明了如何在 API 上表示大数 |

| | | |
|---|---|---|
| 2011 年 4 月 | 1.3-RC8 | 添加了对称加密的数据平面 API，特别是文件 cpa_cy_sym_dp.h。还添加了新类型来表示带有物理寻址的标志缓冲区和缓冲区列表。 |
| 2011 年 4 月 | 1.3-RC9 | 解决以下问题/工作请求:<br><br>· TECG00000098: drbg:澄清了重新播种计数器的描述。<br><br>· TECG00000108: keygen:更新了 MGF 函数的描述，以引用 PKCS#1 MGF1 函数。还在一些 Doxygen 注释中添加了@ref 来美化文档。<br><br>· TECG00000101: nrbg:阐明了请求熵的长度必须大于 0<br><br>· TECG00000097: prime:更新了支持的素数候选位大小列表<br><br>· TECG00000117:更新了 GCM 和 CCM 的各个字段的描述，特别是允许这些算法完全在 API 下实现，从而使实现能够通过 CAVP 的 FIPS 认证 |

| | | |
|---|---|---|
| | | *Note: Data Plane API has been removed from this revision, updates based on previous review and this review will be incorporated in the next revision of the API.* |
| Sep 2010 | 1.3-RC7 | Resolves the following issues/work requests:<br><br>· TECG00000086, “DH API constraints on exponent need to be clarified” – removed offending sentences<br><br>· TECG00000090, “Consider making some CY stats use 64-bit counters” – deprecated 32-bit counters on “legacy” APIs, added 64-bit counter support everywhere<br><br>· Added a symmetric-specific “capability” to specify whether partial packets are supported on a given API instance/implementation |
| Mar 2010 | 1.3-RC5 | Documents version 1.3 Release Candidate #5 of the API, incorporating feedback from the formal review. Key changes:<br><br>· Removed point compression API (pending requirement)<br><br>· Updated DSA API with support for FIPS 186-3<br><br>· Made DRBG reseed function asynchronous and clarified context constraints on this API<br><br>· Numerous other minor clean-ups, clarifications, etc.<br><br>· Added CPA_STATUS_UNSUPPORTED return code to the base API, to be returned when an implementation does not support a given capability. |
| Mar 2010 | 1.3-RC6 | Corrected signature of DRBG session init function to include separate callback function pointers for Generate and Reseed functionality. Also tidied up this revision history table. |
| Dec 2009 | 1.3-RC4 | Documents version 1.3 Release Candidate #4 of the API |

| | | |
|---|---|---|
| | | · TECG00000068: Merged minor changes from EP80579<br><br>· TECG00000069: ECDSA verify – removed input parameter<br><br>· TECG00000047: Updated DSA to support FIPS 186-3<br><br>· TECG00000048: MGF hash function now configurable<br><br>· TECG00000050: Added point decompaction to Elliptic Curve API<br><br>· TECG00000062: Corrected comment re "authenticated cipher" on session setup data structure<br><br>· TECG00000066: Clarified that partial packet is not supported for Kasumi & SNOW3G<br><br>· TECG00000067: Clarified documentation of digestResultLenInBytes<br><br>· TECG00000076: Clarified that for GCM/CCM decrypt, digestVerify is ignored<br><br>*注意:数据平面 API 已从本版本中删除，更新基于之前的审查，本次审查将纳入 API 的下一版本。* |
| 2010 年 9 月 | 1.3-RC7 | 解决以下问题/工作请求:<br><br>· TECG00000086，“DH API 对指数的约束需要澄清”——删除了违规句子<br><br>· TECG00000090，“考虑让一些 CY stats 使用 64 位计数器”-在“遗留”API 上弃用 32 位计数器，在任何地方都添加了 64 位计数器支持<br><br>· 添加了特定于对称的“功能”,以指定在给定的 API 实例/实现上是否支持部分数据包 |
| 2010 年 3 月 | 1.3-RC5 | 记录 API 1.3 版候选版本#5，纳入正式评审的反馈。主要变化:<br><br>· 移除点压缩 API(待定要求)<br><br>· 更新了 DSA API，支持 FIPS 186-3<br><br>· 使 DRBG 重新播种函数异步，并澄清了此 API 上的上下文约束<br><br>· 许多其他小的清理、澄清等。<br><br>· 将 CPA_STATUS_UNSUPPORTED 返回代码添加到基本 API，当实现不支持给定功能时将返回该代码。 |
| 2010 年 3 月 | 1.3-RC6 | 更正了 DRBG 会话初始化函数的签名，以包括用于生成和重新播种功能的单独回调函数指针。还整理了一下这个修订历史表。 |

| | | |
|---|---|---|
| 2009 年 12 月 | 1.3-RC4 | 记录 API 版本 1.3 候选版本 4<br>· TECG00000068:合并了 EP80579 中的微小更改<br>· TECG00000069: ECDSA 验证-已删除输入参数<br>· TECG00000047:更新 DSA 以支持 FIPS 186-3<br>· TECG00000048: MGF 散列函数现在可配置<br>· TECG00000050:向椭圆曲线 API 添加了点分解<br>· TECG00000062:更正了会话设置数据结构中关于“认证密码”的注释<br>· TECG00000066:澄清了小霞& SNOW3G 不支持部分数据包<br>· TECG00000067:澄清了 digestResultLenInBytes 的文档<br>· TECG00000076:澄清了对于 GCM/CCM 解密，digestVerify 将被忽略 |
| | | · TECG00000081: Updated DRBG and NRBG APIs based on feedback from Hifn<br>· TECG00000085: Resolve tech pubs feedback on QA CY API v1.3-RC3 |
| Sep 2009 | 1.3-RC3 | Documents version 1.3 Release Candidate #3 of the API, incorporating feedback from the formal review. Key changes:<br>· On the RBG API, renamed a DRBG “instance” to a “session” (to avoid confusion with other instances and for consistency with symmetric sessions). Also fixed signature of the reseed function, and clarified some comments.<br>· For elliptic curve crypto, clarified some comments.<br>· Made crypto capabilities more granular.<br>· Fixed some @context tags.<br>· Fixed some typos in doxygen @ref tags.<br>· Marked all deprecated functions/types so that they generate warnings when used.<br>· Fixed definitions of TRUE and FALSE.<br>· Added extern “C” linkage to all header files for C++ compilers.<br>· Replaced all tabs with spaces for consistent indentation. |

| | | |
|---|---|---|
| July 2009 | 1.3-RC2 | Documents version 1.3 Release Candidate #2 of the API<br><br>· Base API updated to reflect the decisions around Instances<br><br>· Incorporates feedback from the informal review of v1.3-RC1<br><br>· TECG37: Clarified parameter usage for RSA KeyGen<br><br>· TECG11: Clarified documentation around the enum CpaCyKeyTlsOp |
| June 2009 | 1.3-RC1 | Documents version 1.3 Release Candidate #1 of the API<br><br>· Incorporates the new cipher and authentication algorithms for wireless (Kasumi F8/F9, SNOW3G UEA2/UIA2, AES-CMAC). This was inherited from engineering with minor changes (addition of AES-CMAC, renaming of KGCORE to F8, etc.).<br><br>· TECG17, TECG27: Incorporates the new elliptic curve algorithms. This was inherited from engineering with some minor changes (removed review comments/resolutions, renamed field types, etc.)<br><br>· TECG29: Incorporates the changes to DRBG/NRBG to allow for certification. The old random APIs have been deprecated.<br><br>· TECG25: Adds “capabilities”. Two levels are added: one to indicate which sub-API groups are supported; and for symmetric, one to say which “optional” ciphers are supported.<br><br>· TECG00000081:根据 Hifn 的反馈更新了 DRBG 和 NRBG APIs<br><br>· TECG00000085:解决技术发布者对 QA CY API v1.3-RC3 的反馈 |
| 2009 年 9 月 | 1.3-RC3 | 记录 API 1.3 版候选版本 3，纳入正式评审的反馈。主要变化:<br><br>· 在 RBG API 上，将 DRBG“实例”重命名为“会话”(以避免与其他实例混淆，并与对称会话保持一致)。还修正了补种函数的签名，并澄清了一些评论。<br><br>· 对于椭圆曲线加密，澄清了一些评论。<br><br>· 使加密功能更加精细。<br><br>· 修复了一些@context 标签。<br><br>· 修正了 doxygen @ref 标签中的一些错别字。<br><br>· 标记所有不推荐使用的函数/类型，以便它们在使用时生成警告。<br><br>· 固定的真和假的定义。<br><br>· 为 C++编译器的所有头文件添加了外部“C”链接。<br><br>· 将所有制表符替换为空格，以实现一致的缩进。 |

| | | |
|---|---|---|
| 2009 年 7 月 | 1.3-RC2 | 记录 API 版本 1.3 候选版本 2<br>· 更新了基本 API，以反映围绕实例的决策<br>· 纳入了对 1.3 版-RC1 的非正式审查的反馈<br>· TECG37:阐明了 RSA KeyGen 的参数用法<br>· TECG11:澄清了关于 enum CpaCyKeyTlsOp 的文档 |
| 2009 年 6 月 | 1.3-RC1 | 记录 API 1.3 版候选版本#1<br>· 整合了新的无线加密和认证算法(小霞 F8/F9、SNOW3G UEA2/UIA2、AES-CMAC)。这是从工程继承而来的，稍有改动(增加了 AES-CMAC，将 KGCORE 更名为 F8，等等)。).<br>· TECG17、TECG27:集成了新的椭圆曲线算法。这是从工程继承而来的，有一些小的变化(删除了审核意见/决议，重命名了字段类型等)。)<br>· TECG29:纳入了 DRBG/NRBG 的变更，以允许认证。旧的随机 API 已被弃用。<br>· Merged some changes due to IXA WRs: all comment changes (e.g. addition of RETRY return status from QueryStats functions on some APIs, and other minor clarification text. |
| July 2008 | 1.1 | First released version of this document.<br>Documents version 1.1 of the API. |
| | | 由于 IXA WRs 合并了一些更改:所有注释更改(例如，在一些 API 上添加了来自 QueryStats 函数的重试返回状态，以及其他次要的澄清文本。 |
| 2008 年 7 月 | 1.1 | 本文档的首个发布版本。API 1.1 版的文档。 |

# Table of Contents

# Table of Contents

# Table of Contents

# Table of Contents

# Table of Contents

# Table of Contents

# Table of Contents

# Table of Contents

# Table of Contents

# Table of Contents

# Table of Contents

# 1 Deprecated List
# 1 个不赞成使用的列表

Class **_CpaCyDhStats**
Class _CpaCyDhStats
As of v1.3 of the Crypto API, this structure has been deprecated, replaced by **CpaCyDhStats64**.
从 Crypto API 的 1.3 版开始，这种结构已被取代，由 CpaCyDhStats64

Class **_CpaCyDsaStats**

Class _CpaCyDsaStats
As of v1.3 of the Crypto API, this structure has been deprecated, replaced by **CpaCyDsaStats64**.
从 Crypto API 的 1.3 版开始，这种结构已被取代，由 CpaCyDsaStats64

Class **_CpaCyKeyGenStats**

Class _CpaCyKeyGenStats
As of v1.3 of the Crypto API, this structure has been deprecated, replaced by **CpaCyKeyGenStats64**.
从 Crypto API 的 1.3 版开始，这种结构已被取代，由 CpaCyKeyGenStats64

Class **_CpaCyLnStats**

Class _ CpaCyLnStats
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

Class **_CpaCyPrimeStats**

Class _CpaCyPrimeStats
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

Class **_CpaCyRandGenOpData**

Class _CpaCyRandGenOpData
As of v1.3 of the API, replaced by **CpaCyDrbgGenOpData**.
自 API 1.3 版起，由以下内容取代 CpaCyDrbgGenOpData

Class **_CpaCyRandSeedOpData**

Class _CpaCyRandSeedOpData

As of v1.3 of the API, replaced by **CpaCyDrbgReseedOpData**.
自 API 1.3 版起，由以下内容取代 CpaCyDrbgReseedOpData

Class **_CpaCyRandStats**

Class _CpaCyRandStats
As of v1.3 of the API, replaced by **CpaCyDrbgStats64**.
自 API 1.3 版起，由以下内容取代 CpaCyDrbgStats64

Class **_CpaCyRsaStats**

Class _ CpaCyRsaStats
As of v1.3 of the Crypto API, this structure has been deprecated, replaced by **CpaCyRsaStats64**.
从 Crypto API 的 1.3 版开始，这种结构已被取代，由 CpaCyRsaStats64

Class **_CpaCySymStats**

Class _CpaCySymStats
As of v1.3 of the cryptographic API, this structure has been deprecated, replaced by
从加密 API 的 1.3 版开始，这种结构已被取代，由
**CpaCySymStats64**.
CpaCySymStats64。

Class **_CpaInstanceInfo**

Class _CpaInstanceInfo
As of v1.3 of the Crypto API, this structure has been deprecated, replaced by CpaInstanceInfo2.
从 Crypto API 的 v1.3 开始，这种结构已被弃用，由 CpaInstanceInfo2 取代。

Global **CPA_DEPRECATED**

全球的 CPA_DEPRECATED
As of v1.3 of the Crypto API, this enum has been deprecated, replaced by
从 Crypto API 1.3 版开始，此枚举已被取代，由
**CpaAccelerationServiceType**.
CpaAccelerationServiceType。

Global **CPA_DEPRECATED**

全球的 CPA_DEPRECATED
As of v1.3 of the Crypto API, this enum has been deprecated, replaced by **CpaOperationalState**.
从 Crypto API 1.3 版开始，此枚举已被取代，由 CpaOperationalState

1 Deprecated List

1 个不赞成使用的列表

Global **cpaCyInstanceGetInfo**

全球的 cpaCyInstanceGetInfo

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
**cpaCyInstanceGetInfo2**.
cpaCyInstanceGetInfo2。

Global **cpaCySymQueryStats**

全球的 cpaCySymQueryStats

As of v1.3 of the cryptographic API, this function has been deprecated, replaced by
从加密 API 的 1.3 版开始，此函数已被弃用，由
**cpaCySymQueryStats64()**.
cpaCySymQueryStats64()。

Global **cpaCyKeyGenQueryStats**

全球的 cpaCyKeyGenQueryStats

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
**cpaCyKeyGenQueryStats64()**.
cpaCyKeyGenQueryStats64()。

Global **cpaCyRsaQueryStats**

全球的 cpaCyRsaQueryStats

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
**cpaCyRsaQueryStats64()**.
cpaCyRsaQueryStats64()。

Global **cpaCyDhQueryStats**

全球的 cpaCyDhQueryStats

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
**cpaCyDhQueryStats64()**.
cpaCyDhQueryStats64()。

Global **cpaCyDsaQueryStats**

全球的 cpaCyDsaQueryStats

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()。

Global **cpaCyLnStatsQuery**

全球的 cpaCyLnStatsQuery

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()。

Group **cpaCyRand**

组 cpaCyRand

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** .
Deterministic Random Bit Generation API 和 Non-Deterministic Random Bit Generation API

Global **cpaCyRandGen**

全球的 cpaCyRandGen

As of v1.3 of the API, replaced by **cpaCyDrbgGen()**.
自 API 1.3 版起，由以下内容取代 cpaCyDrbgGen()

Global **cpaCyRandSeed**

全球的 cpaCyRandSeed

As of v1.3 of the API, replaced by **cpaCyDrbgReseed()**.
自 API 1.3 版起，由以下内容取代 cpaCyDrbgReseed()

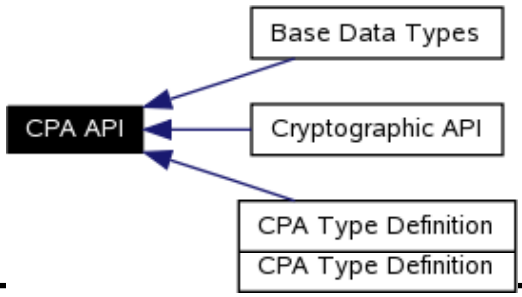
Global **cpaCyRandQueryStats**

全球的 cpaCyRandQueryStats

As of v1.3 of the API, replaced by **cpaCyDrbgQueryStats64()**.
自 API 1.3 版起，由以下内容取代 cpaCyDrbgQueryStats64()

# 2 CPA

Collaboration diagram for CPA API:
CPA API 的协作图:




## 2.1 Detailed Description
## 2.2 详细描述

**File: cpa.h**

文件:cpa.h

This is the top level API definition for Intel(R) QuickAssist Technology. It contains structures, data types and definitions that are common across the interface.
这是英特尔 QuickAssist 技术的顶级 API 定义。它包含了接口中通用的结构、数据类型和定义。

## 2.3 Modules
## 2.4 模块

- **Base Data Types**

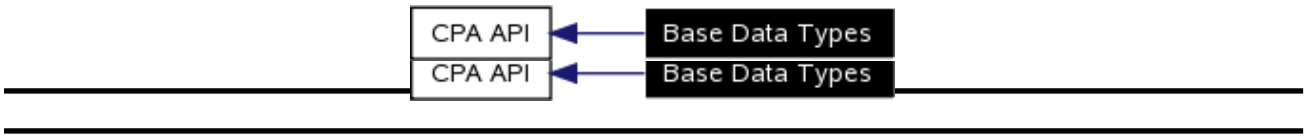
- Base Data Types
- **CPA Type Definition**
- CPA Type Definition
- **Cryptographic API**
- Cryptographic API

# 3 Base Data

**[CPA API]**

[CPA API]

Collaboration diagram for Base Data Types:
基本数据类型的协作图:




## 3.1 Detailed Description
## 3.2 详细描述

**File: cpa.h**

文件:cpa.h

The base data types for the Intel CPA API.
英特尔 CPA API 的基本数据类型。

## 3.3 Data Structures
## 3.4 数据结构

- struct _**CpaFlatBuffer**

- 结构体_CpaFlatBuffer
- struct _**CpaBufferList**
- 结构体_CpaBufferList
- struct _**CpaPhysFlatBuffer**
- 结构体_CpaPhysFlatBuffer
- struct _**CpaPhysBufferList**
- 结构体_CpaPhysBufferList
- struct _**CpaInstanceInfo**
- 结构体_CpaInstanceInfo
- struct _**CpaPhysicalInstanceId**
- 结构体_CpaPhysicalInstanceId
- struct _**CpaInstanceInfo2**
- 结构体_CpaInstanceInfo2

## 3.5 Defines

## 3.6 界定

- #define **CPA_INSTANCE_HANDLE_SINGLE**

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

## 3.4 Typedefs

## 3.5 类型定义

- typedef void * **CpaInstanceHandle**

- typedef void *CpaInstanceHandle
- typedef **Cpa64U CpaPhysicalAddr**
- 数据类型说明Cpa64U CpaPhysicalAddr
- typedef **CpaPhysicalAddr**(* **CpaVirtualToPhysical** )(void *pVirtualAddr)
- 数据类型说明CpaPhysicalAddrCpaVirtualToPhysical
- typedef **_CpaFlatBuffer CpaFlatBuffer**
- 数据类型说明_CpaFlatBuffer CpaFlatBuffer
- typedef **_CpaBufferList CpaBufferList**
- 数据类型说明_CpaBufferList CpaBufferList
- typedef **_CpaPhysFlatBuffer CpaPhysFlatBuffer**
- 数据类型说明_CpaPhysFlatBuffer CpaPhysFlatBuffer
- typedef **_CpaPhysBufferList CpaPhysBufferList**
- 数据类型说明_CpaPhysBufferList CpaPhysBufferList
- typedef **Cpa32S CpaStatus**
- 数据类型说明Cpa32S CpaStatus
- typedef enum **_CpaInstanceType CPA_DEPRECATED**
- typedef 枚举_CpaInstanceType CPA_DEPRECATED
- typedef enum **_CpaAccelerationServiceType CpaAccelerationServiceType**
- typedef 枚举_CpaAccelerationServiceType CpaAccelerationServiceType
- typedef enum **_CpaInstanceState CPA_DEPRECATED**
- typedef 枚举_CpaInstanceState CPA_DEPRECATED
- typedef enum **_CpaOperationalState CpaOperationalState**
- typedef 枚举_CpaOperationalState CpaOperationalState
- typedef **_CpaInstanceInfo CPA_DEPRECATED**
- 数据类型说明_CpaInstanceInfo CPA_DEPRECATED
- typedef **_CpaPhysicalInstanceId CpaPhysicalInstanceId**
- 数据类型说明_CpaPhysicalInstanceId CpaPhysicalInstanceId
- typedef **_CpaInstanceInfo2 CpaInstanceInfo2**
- 数据类型说明_CpaInstanceInfo2 CpaInstanceInfo2
- typedef enum **_CpaInstanceEvent CpaInstanceEvent**
- typedef 枚举_CpaInstanceEvent CpaInstanceEvent

## 3.6 Enumerations
## 3.7 列举

- enum **_CpaInstanceType** {
  **CPA_INSTANCE_TYPE_CRYPTO**,
  **CPA_INSTANCE_TYPE_DATA_COMPRESSION**,
  **CPA_INSTANCE_TYPE_RAID**,
  **CPA_INSTANCE_TYPE_XML**,
  **CPA_INSTANCE_TYPE_REGEX**

- 列举型别_CpaInstanceType
  CPA_INSTANCE_TYPE_CRYPTOCPA_INSTANCE_TYPE_DAT
  A_COMPRESSIONCPA_INSTANCE_TYPE_RAIDCPA_INSTAN
  CE_TYPE_XMLCPA_INSTANCE_TYPE_REGEX
  }

}
- enum **_CpaAccelerationServiceType** {
  **CPA_ACC_SVC_TYPE_CRYPTO**,
  **CPA_ACC_SVC_TYPE_DATA_COMPRESSION**,
  **CPA_ACC_SVC_TYPE_PATTERN_MATCH**,
  **CPA_ACC_SVC_TYPE_RAID**,
  **CPA_ACC_SVC_TYPE_XML**,
  **CPA_ACC_SVC_TYPE_VIDEO_ANALYTICS**
- 列举型别_CpaAccelerationServiceType
  CPA_ACC_SVC_TYPE_CRYPTOCPA_ACC_SVC_TYPE_DATA
  _COMPRESSIONCPA_ACC_SVC_TYPE_PATTERN_MATCHCP
  A_ACC_SVC_TYPE_RAIDCPA_ACC_SVC_TYPE_XMLCPA_A
  CC_SVC_TYPE_VIDEO_ANALYTICS
  **}**
  }
- enum **_CpaInstanceState** {
  **CPA_INSTANCE_STATE_INITIALISED**,
  **CPA_INSTANCE_STATE_SHUTDOWN**
- 列举型别_CpaInstanceState
  CPA_INSTANCE_STATE_INITIALISEDCPA_IN
  STANCE_STATE_SHUTDOWN
  **}**
  }
- enum **_CpaOperationalState** {
  **CPA_OPER_STATE_DOWN**,
  **CPA_OPER_STATE_UP**
- 列 举 型 别
  _CpaOperationalState
  CPA_OPER_STATE_DOWNCPA_OPE
  R_STATE_UP
  **}**
  }
- enum **_CpaInstanceEvent** {
  **CPA_INSTANCE_EVENT_RESTARTING**,
  **CPA_INSTANCE_EVENT_RESTARTED**,
  **CPA_INSTANCE_EVENT_FATAL_ERROR**
- 列举型别_CpaInstanceEvent
  CPA_INSTANCE_EVENT_RESTARTINGCPA_INSTAN
  CE_EVENT_RESTARTEDCPA_INSTANCE_EVENT_FA
  TAL_ERROR
  **}**
  }

---

## 3.8 Data Structure Documentation

## 3.9 数据结构文档

## 3.6.1 _CpaFlatBuffer Struct Reference

## 3.6.2 _CpaFlatBuffer 结构引用

### 3.6.2.1 Detailed Description
### 3.6.2.2 详细描述

Flat buffer structure containing a pointer and length member.
包含指针和长度成员的平面缓冲区结构。

A flat buffer structure. The data pointer, pData, is a virtual address. An API instance may require the actual data to be in contiguous physical memory as determined by **CpaInstanceInfo2**.
扁平缓冲结构。数据指针 pData 是一个虚拟地址。API 实例可能要求实际数据位于连续的物理内存中，这由 CpaInstanceInfo2

### 3.6.2.3 Data Fields
### 3.6.2.4 数据字段

- **Cpa32U dataLenInBytes**
- Cpa32U dataLenInBytes
- **Cpa8U * pData**
- Cpa8U *pData

### 3.6.2.5 Field Documentation
### 3.6.2.6 现场文件

**Cpa32U _CpaFlatBuffer::dataLenInBytes**

Data length specified in bytes. When used as an input parameter to a function, the length specifies the current length of the buffer. When used as an output parameter to a function, the length passed in specifies
Cpa32_CpaFlatBuffer::dataLenInByte
the maximum length of the buffer on return (i.e. the allocated length). The implementation will not write past
返回时缓冲区的最大长度(即分配的长度)。实现不会重写
this length. On return, the length is always unchanged.
这个长度。返回时，长度始终不变。

**Cpa8U* _CpaFlatBuffer::pData**

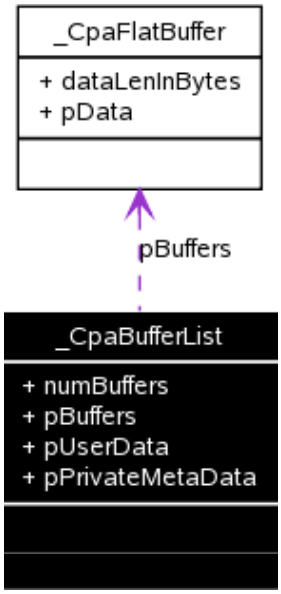
The data pointer is a virtual address, however the actual data pointed to is required to be in contiguous physical memory unless the field requiresPhysicallyContiguousMemory in CpaInstanceInfo2 is false.
Cpa8U_CpaFlatBuffer::pDat

## 3.6.3 _CpaBufferList Struct Reference

### 3.6.4 _CpaBufferList 结构引用

Collaboration diagram for _CpaBufferList:

_CpaBufferList 的协作图:



3.6.2 _CpaBufferList Struct Reference

3.6.3_CpaBufferList 结构引用

### 3.6.3.1 Detailed Description

### 3.6.3.2 详细描述

Scatter/Gather buffer list containing an array of flat buffers.
包含平面缓冲区数组的分散/聚集缓冲区列表。

A scatter/gather buffer list structure. This buffer structure is typically used to represent a region of memory which is not physically contiguous, by describing it as a collection of buffers, each of which is physically contiguous.
分散/聚集缓冲区列表结构。这种缓冲区结构通常用于表示物理上不连续的内存区域，方法是将其描述为一组缓冲区，每个缓冲区都是物理上连续的。

**Note:**
**注意：**

The memory for the pPrivateMetaData member must be allocated by the client as physically contiguous memory. When allocating memory for pPrivateMetaData, a call to the corresponding BufferListGetMetaSize function (e.g. cpaCyBufferListGetMetaSize) MUST be made to determine the size of the Meta Data Buffer. The returned size (in bytes) may then be passed in a memory allocation routine to allocate the pPrivateMetaData memory.
pPrivateMetaData 成员的内存必须由客户端分配为物理上连续的内存。为 pPrivateMetaData 分配内存时，必须调用相应的 BufferListGetMetaSize 函数(例如 cpaCyBufferListGetMetaSize)来确定元数据缓冲区的大小。然后，返回的大小(以字节为单位)可以在内存分配例程中传递，以分配 pPrivateMetaData 内存。

### 3.6.3.3 Data Fields
### 3.6.3.4 数据字段

- **Cpa32U numBuffers**
- Cpa32U numBuffers
- **CpaFlatBuffer * pBuffers**
- CpaFlatBuffer *pBuffers
- void * **pUserData**
- 无效*pUserData
- void * **pPrivateMetaData**
- 无效*pPrivateMetaData

### 3.6.3.5 Field Documentation
### 3.6.3.6 现场文件

**Cpa32U CpaBufferList::numBuffers**
Number of buffers in the list
列表中的缓冲区数量

**CpaFlatBuffer* CpaBufferList::pBuffers**
Pointer to an unbounded array containing the number of CpaFlatBuffers defined by numBuffers
指向一个无界数组的指针，该数组包含由 numBuffers 定义的 CpaFlatBuffers 的数量

void* **CpaBufferList::pUserData**
Reference to the client data
对客户数据的引用

This is an opaque field that is not read or modified internally.
这是一个不透明的字段，不能在内部读取或修改。

void* **CpaBufferListupPrivateMetaData**

Private representation of this buffer list. The memory for this buffer needs to be allocated by the client as contiguous data. The amount of memory required is returned with a call to the corresponding BufferListGetMetaSize function. If that function returns a size of zero then no memory needs to be allocated, and this parameter can be NULL.
此缓冲区列表的私有表示。客户端需要将该缓冲区的内存作为连续数据进行分配。通过调用相应的 BufferListGetMetaSize 函数返回所需的内存量。如果该函数返回的大小为零，则不需要分配内存，并且该参数可以为 NULL。

---

## 3.6.4 _CpaPhysFlatBuffer Struct Reference

## 3.6.5 _CpaPhysFlatBuffer 结构引用

### 3.6.5.1 Detailed Description
### 3.6.5.2 详细描述

Flat buffer structure with physical address.
具有物理地址的平面缓冲结构。

Functions taking this structure do not need to do any virtual to physical address translation before writing the buffer to hardware.
采用这种结构的函数在将缓冲区写入硬件之前不需要进行任何虚拟到物理地址的转换。

### 3.6.5.3 Data Fields
### 3.6.5.4 数据字段

- **Cpa32U dataLenInBytes**
- Cpa32U dataLenInBytes
- **Cpa32U reserved**
- Cpa32U reserved

3.6.3 _CpaPhysFlatBuffer Struct Reference

3.6.4_CpaPhysFlatBuffer 结构引用

- **CpaPhysicalAddr bufferPhysAddr**
  - CpaPhysicalAddr bufferPhysAddr

**3.6.5.5 Field Documentation**
**3.6.5.6 现场文件**

Cpa32U _CpaPhysFlatBuffer::dataLenInBytes

Data length specified in bytes. When used as an input parameter to a function, the length specifies the current length of the buffer. When used as an output parameter to a function, the length passed in specifies the maximum length of the buffer on return (i.e. the allocated length). The implementation will not write past this length. On return, the length is always unchanged.
以字节指定的数据长度。当用作函数的输入参数时，长度指定缓冲区的当前长度。当用作函数的输出参数时，传入的长度指定返回时缓冲区的最大长度(即分配的长度)。实现不会写入超过这个长度。返回时，长度始终不变。

Cpa32U _CpaPhysFlatBuffer::reserved

Reserved for alignment
保留用于对齐

CpaPhysicalAddr _CpaPhysFlatBuffer::bufferPhysAddr

The physical address at which the data resides. The data pointed to is required to be in contiguous physical memory.
数据驻留的物理地址。指向的数据需要在连续的物理内存中。

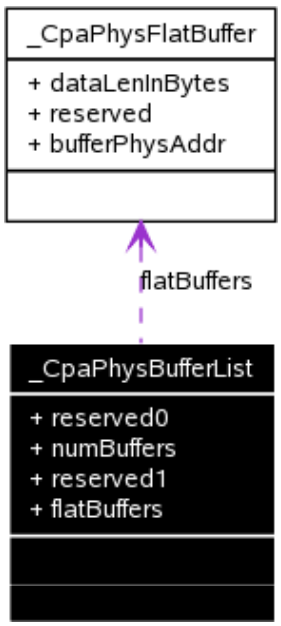
---

## 3.6.5 _CpaPhysBufferList Struct Reference

## 3.6.6 _CpaPhysBufferList 结构引用

Collaboration diagram for _CpaPhysBufferList:

_CpaPhysBufferList 的协作图:



#### 3.6.4.1 Detailed Description
#### 3.6.4.2 详细描述

Scatter/gather list containing an array of flat buffers with physical addresses.
包含具有物理地址的平面缓冲区数组的分散/收集列表。

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

3.6.4 _CpaPhysBufferList Struct Reference

3.6.5_CpaPhysBufferList 结构引用

**3.6.4.3 Data Fields**

**3.6.4.4 数据字段**

- **Cpa64U reserved0**
- Cpa64U reserved0
- **Cpa32U numBuffers**
- Cpa32U numBuffers
- **Cpa32U reserved1**
- Cpa32U reserved1
- **CpaPhysFlatBuffer flatBuffers []**
- CpaPhysFlatBuffer flatBuffers []

**3.6.4.5 Field Documentation**

**3.6.4.6 现场文件**

Reserved for internal usage

保留供内部使用

Number of buffers in the list

列表中的缓冲区数量

Reserved for alignment

保留用于对齐

Array of flat buffer structures, of size numBuffers

大小为 numBuffers 的平面缓冲区结构数组

---

## 3.6.6 _CpaInstanceInfo Struct Reference

## 3.6.7 _CpaInstanceInfo 结构引用

**3.6.7.1 Detailed Description**

Instance Info Structure

**Deprecated:**

**3.6.7.2** 实例信息结构的详细

描述 Deprecated:

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

从 Crypto API 的 v1.3 开始，这种结构已被弃用，由 CpaInstanceInfo2 取代。

Structure that contains the information to describe the instance.

结构，它包含描述实例的信息。

### 3.6.7.3 Data Fields
### 3.6.7.4 数据字段

- enum _**CpaInstanceType type**
- 列举型别_CpaInstanceType type
- enum _**CpaInstanceState state**
- 列举型别_CpaInstanceState state
- **Cpa8U name** [CPA_INSTANCE_MAX_NAME_SIZE_IN_BYTES]
- Cpa8U name［注册会计师实例最大名称大小字节］
- **Cpa8U version** [CPA_INSTANCE_MAX_VERSION_SIZE_IN_BYTES]
- Cpa8U version［注册会计师实例最大版本大小字节］

### 3.6.7.5 Field Documentation
### 3.6.7.6 现场文件

Type definition for this instance.
此实例的类型定义。

Operational state of the instance.
实例的运行状态。

Simple text string identifier for the instance.
实例的简单文本字符串标识符。

3.6.5 _CpaInstanceInfo Struct Reference

3.6.6_CpaInstanceInfo 结构引用

Version string. There may be multiple versions of the same type of instance accessible through a particular library.
版本字符串。可以通过特定的库访问同一类型实例的多个版本。

---

## 3.6.7 _CpaPhysicalInstanceId Struct Reference

## 3.6.8 _ CpaPhysicalInstanceId 结构引用

### 3.6.8.1 Detailed Description
### 3.6.8.2 详细描述

Physical Instance ID
物理实例 ID

Identifies the physical instance of an accelerator execution engine.
标识加速器执行引擎的物理实例。

Accelerators grouped into "packages". Each accelerator can in turn contain one or more execution engines. Implementations of this API will define the packageId, acceleratorId, executionEngineId and busAddress as appropriate for the implementation. For example, for hardware-based accelerators, the packageId might identify the chip, which might contain multiple accelerators, each of which might contain multiple execution engines. The combination of packageId, acceleratorId and executionEngineId uniquely identifies the instance.
归入“包”的加速器。每个加速器又可以包含一个或多个执行引擎。该 API 的实现将为实现定义适当的 packageId、acceleratorId、executionEngineId 和 busAddress。例如，对于基于硬件的加速器，packageId 可以标识芯片，该芯片可以包含多个加速器，每个加速器可以包含多个执行引擎。packageId、acceleratorId 和 executionEngineId 的组合唯一标识该实例。

Hardware based accelerators implementing this API may also provide information on the location of the accelerator in the busAddress field. This field will be defined as appropriate for the implementation. For example, for PCIe attached accelerators, the busAddress may contain the PCIe bus, device and function number of the accelerators.
实现该 API 的基于硬件的加速器也可以在总线地址字段中提供关于加速器位置的信息。该字段将被定义为适用于实施。例如，对于 PCIe 附属加速器，总线地址可以包含加速器的 PCIe 总线、设备和功能号。

### 3.6.8.3 Data Fields
### 3.6.8.4 数据字段

- **Cpa16U packageId**
- Cpa16U packageId
- **Cpa16U acceleratorId**
- Cpa16U acceleratorId
- **Cpa16U executionEngineId**
- Cpa16U executionEngineId
- **Cpa16U busAddress**
- Cpa16U busAddress
- **Cpa32U kptAcHandle**

- Cpa32U kptAcHandle

### 3.6.8.5 Field Documentation
### 3.6.8.6 现场文件

Cpa16U CpaPhysicalInstanceId::packageId

Identifies the package within which the accelerator is contained.
标识包含加速器的包。

Cpa16U CpaPhysicalInstanceId::acceleratorId

Identifies the specific accelerator within the package.
标识包中的特定加速器。

Cpa16U CpaPhysicalInstanceId::executionEngineId

Identifies the specific execution engine within the accelerator.
标识加速器中的特定执行引擎。

Cpa16U CpaPhysicalInstanceId::busAddress

Identifies the bus address associated with the accelerator execution engine.
标识与加速器执行引擎相关联的总线地址。

Cpa32U CpaPhysicalInstanceId::kptAcHandle

Identifies the achandle of the accelerator.
识别加速器的手柄。

3.6.6 _CpaPhysicalInstanceId Struct Reference

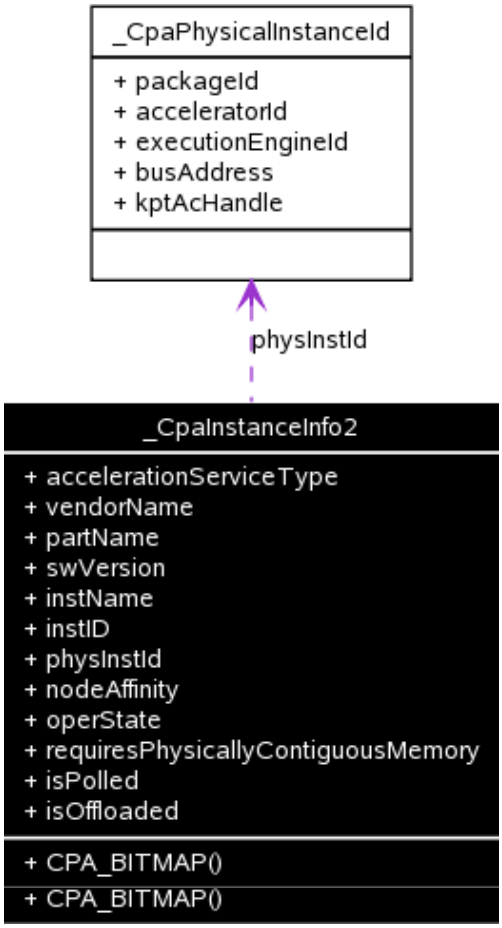
3.6.7_ CpaPhysicalInstanceId 结构引用

## 3.6.8 _CpaInstanceInfo2 Struct Reference

## 3.6.9 _CpaInstanceInfo2 结构引用

Collaboration diagram for _CpaInstanceInfo2:

_CpaInstanceInfo2 的协作图:




### 3.6.9.1 Detailed Description
### 3.6.9.2 详细描述

Instance Info Structure, version 2
实例信息结构，版本 2

Structure that contains the information to describe the instance.
结构，它包含描述实例的信息。

### 3.6.9.3 Public Member Functions

### 3.6.9.4 公共成员函数

- **CPA_BITMAP** (coreAffinity, CPA_MAX_CORES)
- CPA_BITMAP（核心关联性，CPA_MAX_CORES）

### 3.6.9.5 Data Fields
### 3.6.9.6 数据字段

- **CpaAccelerationServiceType accelerationServiceType**
- CpaAccelerationServiceType accelerationServiceType
- **Cpa8U vendorName** [CPA_INST_VENDOR_NAME_SIZE]
- Cpa8U vendorName［注册会计师_ INST _供应商_名称_大小］
- **Cpa8U partName** [CPA_INST_PART_NAME_SIZE]
- Cpa8U partName［注册会计师_ PART _零件_名称_尺寸］
- **Cpa8U swVersion** [CPA_INST_SW_VERSION_SIZE]
- Cpa8U swVersion［注册会计师_ INST _软件_版本_大小］
- **Cpa8U instName** [CPA_INST_NAME_SIZE]
- Cpa8U instName［注册会计师_ INST _名称_大小］
- **Cpa8U instID** [CPA_INST_ID_SIZE]
- Cpa8U instID［注册会计师_INST_ID_SIZE］
- **CpaPhysicalInstanceId physInstId**
- CpaPhysicalInstanceId physInstId
- **Cpa32U nodeAffinity**
- Cpa32U nodeAffinity
- **CpaOperationalState operState**
- CpaOperationalState operState
- **CpaBoolean requiresPhysicallyContiguousMemory**
- CpaBoolean requiresPhysicallyContiguousMemory

3.6.7 _CpaInstanceInfo2 Struct Reference

3.6.8 _CpaInstanceInfo2 结构引用

- **CpaBoolean isPolled**

- CpaBoolean isPolled
- **CpaBoolean isOffloaded**
- CpaBoolean isOffloaded

**3.6.9.7 Member Function Documentation**
**3.6.9.8 成员函数文档**

CpaInstanceInfo2::CPA_BITMAP( coreAffinity
CpaInstanceInfo2::CPA_BITMAP( coreAffinity
CPA_MAX_CORES ,
CPA_MAX_CORES
)

A bitmap identifying the core or cores to which the instance is affinitized in an SMP operating system.
标识 SMP 操作系统中实例关联到的一个或多个核心的位图。

The term core here is used to mean a "logical" core - for example, in a dual-processor, quad-core system with hyperthreading (two threads per core), there would be 16 such cores (2 processors x 4 cores/processor x 2 threads/core). The numbering of these cores and the corresponding bit positions is OS-specific. Note that Linux refers to this as "processor affinity" or "CPU affinity", and refers to the bitmap as a "cpumask".
这里的“核心”一词是指“逻辑”核心，例如，在具有超线程技术(每个核心两个线程)的双处理器、四核系统中，将有 16 个这样的核心(2 个处理器 x 4 个核心/处理器 x 2 个线程/核心)。这些内核的编号和相应的位位置是特定于操作系统的。请注意，Linux 将此称为“处理器关联”或“CPU 关联”，并将位图称为“cpumask”。

The term "affinity" is used to mean that this is the core on which the callback function will be invoked when using the asynchronous mode of the API. In a hardware-based implementation of the API, this might be the core to which the interrupt is affinitized. In a software-based implementation, this might be the core to which the process running the algorithm is affinitized. Where there is no affinity, the bitmap can be set to all zeroes.
术语“亲合性”用于表示当使用 API 的异步模式时，回调函数将在这个核心上被调用。在基于硬件的 API 实现中，这可能是中断所关联的核心。在基于软件的实现中，这可能是运行算法的进程所关联的核心。在没有关联性的情况下，位图可以被设置为全零。

This bitmap should be manipulated using the macros **CPA_BITMAP_BIT_SET**, **CPA_BITMAP_BIT_CLEAR** and **CPA_BITMAP_BIT_TEST**.
这个位图应该使用宏来操作 CPA_BITMAP_BIT_SETCPA_BITMAP_BIT_CLEAR CPA_BITMAP_BIT_TEST

**3.6.9.9 Field Documentation**
**3.6.9.10 现场文件**

CpaAccelerationServiceType CpaInstanceInfo2::accelerationServiceType
CpaAccelerationServiceType CpaInstanceInfo2::accelerationServiceType

Type of service provided by this instance.
此实例提供的服务类型。

Cpa8U CpaInstanceInfo2::vendorName[CPA_INST_VENDOR_NAME_SIZE]
Cpa8U CpaInstanceInfo2::vendorName[CPA_INST_VENDOR_NAME_SIZE]

String identifying the vendor of the accelerator.
标识加速器供应商的字符串。

Cpa8U CpaInstanceInfo2::partName[CPA_INST_PART_NAME_SIZE]
Cpa8U CpaInstanceInfo2::partName[CPA_INST_PART_NAME_SIZE]

String identifying the part (name and/or number).

标识零件的字符串(名称和/或编号)。

String identifying the version of the software associated with the instance. For hardware-based implementations of the API, this should be the driver version. For software-based implementations of the API, this should be the version of the library.
标识与实例关联的软件版本的字符串。对于基于硬件的 API 实现，这应该是驱动程序版本。对于基于软件的 API 实现，这应该是库的版本。

Note that this should NOT be used to store the version of the API, nor should it be used to report the hardware revision (which can be captured as part of the **partName**, if required).
注意，这不应该用于存储 API 的版本，也不应该用于报告硬件版本(可以作为 partName

String identifying the name of the instance.
标识实例名称的字符串。

String containing a unique identifier for the instance
包含实例的唯一标识符的字符串

Identifies the "physical instance" of the accelerator.
标识加速器的“物理实例”。

Identifies the processor complex, or node, to which the accelerator is physically connected, to help identify locality in NUMA systems.
标识加速器物理连接到的处理器组合系统或节点，以帮助标识 NUMA 系统中的位置。

The values taken by this attribute will typically be in the range 0..n-1, where n is the number of nodes (processor complexes) in the system. For example, in a dual-processor configuration, n=2. The precise values and their interpretation are OS-specific.
该属性的取值范围通常为 0..n-1，其中 n 是系统中节点(处理器复合体)的数量。例如，在双处理器配置中，n=2。精确的值及其解释是特定于操作系统的。

Operational state of the instance.
实例的运行状态。

Specifies whether the data pointed to by flat buffers (**CpaFlatBuffer::pData**) supplied to this instance must be in physically contiguous memory.
指定平面缓冲区所指向的数据(CpaFlatBuffer::pData

Specifies whether the instance must be polled, or is event driven. For hardware accelerators, the alternative to polling would be interrupts.
指定实例是必须轮询还是由事件驱动。对于硬件加速器，轮询的替代方案是中断。

Identifies whether the instance uses hardware offload, or is a software-only implementation.
标识实例是使用硬件卸载，还是纯软件实现。

---

## 3.7 Define Documentation

## 3.8 定义文档

Default instantiation handle value where there is only a single instance

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

只有一个实例时的默认实例化句柄值用作只有一个实例时的实例句柄

值。

Special value which can be taken by length fields on some of the "data plane" APIs to indicate that the buffer in question is of type CpaPhysBufferList, rather than simply an array of bytes.
一些“数据平面”API 上的长度字段可以采用的特殊值，用于指示所讨论的缓冲区属于 CpaPhysBufferList 类型，而不仅仅是一个字节数组。

Success status value.

2.7 Define

成功状态值。

#define CRA_STATUS_FAIL

Fail status value.

失败状态值。

#define CRA_STATUS_RETRY

Retry status value.

重试状态值。

#define CRA_STATUS_RESOURCE

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

请求的资源不可用。请参考 API 的相关章节，了解建议行动的具体内容。

#define CRA_STATUS_INVALID_PARAM

Invalid parameter has been passed in.

传入了无效参数。

#define CRA_STATUS_FATAL

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

出现了严重错误。建议的措施是关闭并重新启动组件。

#define CPA_STATUS_UNSUPPORTED

The function is not supported, at least not with the specific parameters supplied. This may be because a particular capability is not supported by the current implementation.
不支持该函数，至少不支持提供的特定参数。这可能是因为当前的实现不支持特定的功能。

#define CPA_STATUS_RESTARTING

The API implementation is restarting. This may be reported if, for example, a hardware implementation is undergoing a reset. Recommended course of action is to retry the request.
API 实现正在重新启动。例如，如果硬件实现正在经历复位，则可能会报告这种情况。建议采取的措施是重试请求。

#define CPA_STATUS_MAX_STR_LENGTH_IN_BYTES

API status string type definition
API 状态字符串类型定义

This type definition is used for the generic status text strings provided by cpaXxGetStatusText API functions. Common values are defined, for example see **CPA_STATUS_STR_SUCCESS**, **CPA_STATUS_FAIL**, etc., as well as the maximum size **CPA_STATUS_MAX_STR_LENGTH_IN_BYTES**.
该类型定义用于 cpaXxGetStatusText API 函数提供的通用状态文本字符串。定义了通用值，例如参见 CPA_STATUS_STR_SUCCESSCPA_STATUS_FAILCPA_STATUS_MAX_STR_LENGTH_IN_BYTES

Maximum length of the Overall Status String (including generic and specific strings returned by calls to cpaXxGetStatusText)
整体状态字符串的最大长度(包括调用 cpaXxGetStatusText 返回的一般和特定字符串)

#define CPA_STATUS_STR_SUCCESS

Status string for **CPA_STATUS_SUCCESS**.
的状态字符串 CPA_STATUS_SUCCESS

#define CPA_STATUS_STR_FAIL

Status string for **CPA_STATUS_FAIL**.
的状态字符串 CPA_STATUS_FAIL

#define CPA_STATUS_STR_RETRY

Status string for **CPA_STATUS_RETRY**.
的状态字符串 CPA_STATUS_RETRY

#define CPA_STATUS_STR_RESOURCE

Status string for **CPA_STATUS_RESOURCE**.
的状态字符串 CPA_STATUS_RESOURCE

#define CPA_STATUS_STR_INVALID_PARAM

Status string for **CPA_STATUS_INVALID_PARAM**.
的状态字符串 CPA_STATUS_INVALID_PARAM

#define CPA_STATUS_STR_FATAL

Status string for **CPA_STATUS_FATAL**.
的状态字符串 CPA_STATUS_FATAL

#define CPA_STATUS_STR_UNSUPPORTED

Status string for **CPA_STATUS_UNSUPPORTED**.
的状态字符串 CPA_STATUS_UNSUPPORTED

#define CPA_INSTANCE_MAX_NAME_SIZE_IN_BYTES

2.7 Define

Maximum instance info name string length in bytes
以字节为单位的最大实例信息名称字符串长度

#define CRA_INSTANCE_MAX_ID_SIZE_IN_BYTES

Maximum instance info id string length in bytes
以字节为单位的最大实例信息 id 字符串长度

#define CRA_INSTANCE_MAX_VERSION_SIZE_IN_BYTES

Maximum instance info version string length in bytes
以字节为单位的最大实例信息版本字符串长度

## 3.9 Typedef Documentation

## 3.10 Typedef 文档

typedef void* **CpaInstanceHandle**

typedef void*CpaInstanceHandl

Instance handle type.

实例句柄类型。

Handle used to uniquely identify an instance.

用于唯一标识实例的句柄。

**Note:**

**注意:**

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

在仅存在单个实例化的情况下，该字段可以被设置为

**CPA_INSTANCE_HANDLE_SINGLE**.

CPA_INSTANCE_HANDLE_SINGLE。

typedef **Cpa64U CpaPhysicalAddr**

数据类型说明 Cpa64CpaPhysicalAdd

Physical memory address.

物理内存地址。

Type for physical memory addresses.

物理内存地址的类型。

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

Virtual to physical address conversion routine.

数据类型说明 CpaPhysicalAddrCpaVirtualToPhysical

(*

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

该函数用于将虚拟地址转换为物理地址。

**Context:**

**背景:**

The function shall not be called in an interrupt context.

该函数不应在中断上下文中调用。

**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] *pVirtualAddr* Virtual address to be converted.
[in]要转换的 pVirtualAddr 虚拟地址。

**Returns:**
**退货:**
Returns the corresponding physical address. On error, the value NULL is returned.
返回相应的物理地址。出错时，返回值 NULL。

**Postcondition:**
**后置条件:**
None
没有人

**See also:**
**另请参见:**
None
没有人

typedef struct _**CpaFlatBuffer CpaFlatBuffer**
typedef 结构_CpaFlatBuffeCpaFlatBuffe

Flat buffer structure containing a pointer and length member.

包含指针和长度成员的平面缓冲区结构。

A flat buffer structure. The data pointer, pData, is a virtual address. An API instance may require the actual data to be in contiguous physical memory as determined by **CpaInstanceInfo2**.
扁平缓冲结构。数据指针 pData 是一个虚拟地址。API 实例可能要求实际数据位于连续的物理内存中，这由 CpaInstanceInfo2

typedef struct _**CpaBufferList CpaBufferList**
typedef 结构_CpaBufferLisCpaBufferLis

Scatter/Gather buffer list containing an array of flat buffers.
包含平面缓冲区数组的分散/聚集缓冲区列表。

A scatter/gather buffer list structure. This buffer structure is typically used to represent a region of memory which is not physically contiguous, by describing it as a collection of buffers, each of which is physically contiguous.
分散/聚集缓冲区列表结构。这种缓冲区结构通常用于表示物理上不连续的内存区域，方法是将其描述为一组缓冲区，每个缓冲区都是物理上连续的。

**Note:**
**注意:**

The memory for the pPrivateMetaData member must be allocated by the client as physically contiguous memory. When allocating memory for pPrivateMetaData, a call to the corresponding BufferListGetMetaSize function (e.g. cpaCyBufferListGetMetaSize) MUST be made to determine the size of the Meta Data Buffer. The returned size (in bytes) may then be passed in a memory allocation routine to allocate the pPrivateMetaData memory.
pPrivateMetaData 成员的内存必须由客户端分配为物理上连续的内存。为 pPrivateMetaData 分配内存时，必须调用相应的 BufferListGetMetaSize 函数(例如 cpaCyBufferListGetMetaSize)来确定元数据缓冲区的大小。然后，返回的大小(以字节为单位)可以在内存分配例程中传递，以分配 pPrivateMetaData 内存。

typedef struct _**CpaPhysFlatBuffer CpaPhysFlatBuffer**
typedef 结构_CpaPhysFlatBuffeCpaPhysFlatBuffe

Flat buffer structure with physical address.
具有物理地址的平面缓冲结构。

Functions taking this structure do not need to do any virtual to physical address translation before writing the buffer to hardware.
采用这种结构的函数在将缓冲区写入硬件之前不需要进行任何虚拟到物理地址的转换。

typedef struct _**CpaPhysBufferList CpaPhysBufferList**
typedef 结构_CpaPhysBufferLisCpaPhysBufferLis

Scatter/gather list containing an array of flat buffers with physical addresses.
包含具有物理地址的平面缓冲区数组的分散/收集列表。

Similar to **CpaBufferList**, this buffer structure is typically used to represent a region of memory which is not physically contiguous, by describing it as a collection of buffers, each of which is

physically contiguous. The difference is that, in this case, the individual "flat" buffers are represented using physical, rather than virtual, addresses.
类似 CpaBufferList

### typedef **Cpa32S CpaStatus**

数据类型说明 Cpa32CpaStatu

API status value type definition
API 状态值类型定义

This type definition is used for the return values used in all the API functions. Common values are defined, for example see **CPA_STATUS_SUCCESS**, **CPA_STATUS_FAIL**, etc.
该类型定义用于所有 API 函数中使用的返回值。定义了通用值，例如参见 CPA_STATUS_SUCCESSCPA_STATUS_FAIL

### typedef enum _**CpaInstanceType CPA_DEPRECATED**

typedef 枚举_CpaInstanceTypCPA_DEPRECATE

Instance Types
实例类型

**Deprecated:**
Deprecated:
As of v1.3 of the Crypto API, this enum has been deprecated, replaced by **CpaAccelerationServiceType**.
从 Crypto API 1.3 版开始，此枚举已被取代，由 CpaAccelerationServiceType。

Enumeration of the different instance types.
不同实例类型的枚举。

### typedef enum _**CpaAccelerationServiceType CpaAccelerationServiceType**

typedef 枚举_CpaAccelerationServiceTypCpaAccelerationServiceTyp

Service Type
通用式

Enumeration of the different service types.
不同服务类型的枚举。

2.8 Typedef

typedef enum CpaInstanceState CPA_DEPRECATED

Instance State
实例状态

**Deprecated:**
Deprecated:
As of v1.3 of the Crypto API, this enum has been deprecated, replaced by **CpaOperationalState**.
从 Crypto API 1.3 版开始，此枚举已被取代，由 CpaOperationalState

Enumeration of the different instance states that are possible.
可能的不同实例状态的枚举。

typedef enum CpaOperationalState CpaOperationalState

Instance operational state
实例操作状态

Enumeration of the different operational states that are possible.
列举可能的不同操作状态。

typedef struct CpaInstanceInfo CPA_DEPRECATED

Instance Info Structure
实例信息结构

**Deprecated:**
Deprecated:
As of v1.3 of the Crypto API, this structure has been deprecated, replaced by CpaInstanceInfo2.
从 Crypto API 的 v1.3 开始，这种结构已被弃用，由 CpaInstanceInfo2 取代。

Structure that contains the information to describe the instance.
结构，它包含描述实例的信息。

typedef struct CpaPhysicalInstanceId CpaPhysicalInstanceId

Physical Instance ID
物理实例 ID

Identifies the physical instance of an accelerator execution engine.
标识加速器执行引擎的物理实例。

Accelerators grouped into "packages". Each accelerator can in turn contain one or more execution engines. Implementations of this API will define the packageId, acceleratorId, executionEngineId and busAddress as appropriate for the implementation. For example, for hardware-based accelerators, the packageId might identify the chip, which might contain multiple accelerators, each of which might contain multiple execution engines. The combination of packageId, acceleratorId and executionEngineId uniquely identifies the instance.
归入“包”的加速器。每个加速器又可以包含一个或多个执行引擎。该 API 的实现将为实现定义适当的 packageId、acceleratorId、executionEngineId 和 busAddress。例如，对于基于硬件的加速器，packageId 可以标识芯片，该芯片可以包含多个加速器，每个加速器可以包含多个执行引擎。

packageId、acceleratorId 和 executionEngineId 的组合唯一标识该实例。

Hardware based accelerators implementing this API may also provide information on the location of the accelerator in the busAddress field. This field will be defined as appropriate for the implementation. For example, for PCIe attached accelerators, the busAddress may contain the PCIe bus, device and function number of the accelerators.
实现该 API 的基于硬件的加速器也可以在总线地址字段中提供关于加速器位置的信息。该字段将被定义为适用于实施。例如，对于 PCIe 附属加速器，总线地址可以包含加速器的 PCIe 总线、设备和功能号。

typedef struct **CpaInstanceInfo2 CpaInstanceInfo2**

Instance Info Structure, version 2
实例信息结构，版本 2

Structure that contains the information to describe the instance.
结构，它包含描述实例的信息。

typedef enum **CpaInstanceEvent CpaInstanceEvent**

Instance Events
实例事件

Enumeration of the different events that will cause the registered Instance notification callback function to be invoked.
将导致调用注册实例通知回调函数的不同事件的枚举。

## 3.11 Enumeration Type Documentation

## 3.12 枚举类型文档

enum **_CpaInstanceType**
列举型别_CpaInstanceTyp

Instance Types
实例类型

**Deprecated:**
Deprecated:
As of v1.3 of the Crypto API, this enum has been deprecated, replaced by **CpaAccelerationServiceType**.
从 Crypto API 1.3 版开始，此枚举已被取代，由 CpaAccelerationServiceType。

Enumeration of the different instance types.
不同实例类型的枚举。

**Enumerator:**
**枚举器:**

| | |
|---|---|
| *CPA_INSTANCE_TYPE_CRYPTO* | Cryptographic instance type |
| *CPA_INSTANCE_TYPE_DATA_COMPRESSION* | Data compression instance type |
| *CPA_INSTANCE_TYPE_RAID* | RAID instance type |
| *CPA_INSTANCE_TYPE_CRYPTO 加密实例类型* | *CPA _ INSTANCE _ TYPE _ DATA _ COMPRESSION 数据压缩实例类型* |
| *CPA_INSTANCE_TYPE_RAID* | RAID 实例类型 |
| *CPA_INSTANCE_TYPE_XML* | XML instance type |
| *CPA_INSTANCE_TYPE_XML XML 实例类型* | |
| *CPA_INSTANCE_TYPE_REGEX* | Regular Expression instance type |
| *CPA_INSTANCE_TYPE_REGEX 正则表达式实例类型* | |

enum **_CpaAccelerationServiceType**
列举型别_CpaAccelerationServiceTyp

Service Type
通用式

Enumeration of the different service types.
不同服务类型的枚举。

**Enumerator:**
**枚举器:**

| | |
|---|---|
| *CPA_ACC_SVC_TYPE_CRYPTO* | Cryptography |
| *CPA _ ACC _ SVC _ TYPE _ Cryptography* | |
| *CPA_ACC_SVC_TYPE_DATA_COMPRESSION* | Data |
| *CPA _ ACC _ SVC _ TYPE _ DATA _压缩数据* | |

Compression
*CPA_ACC_SVC_TYPE_PATTERN_MATCH* Pattern Match
*CPA_ACC_SVC_TYPE_RAID* RAID
压缩 CPA _
ACC _ SVC _ TYPE _模式_匹配 模式匹配
CPA_ACC_SVC_TYPE_RAID 袭击
*CPA_ACC_SVC_TYPE_XML* XML
*CPA_ACC_SVC_TYPE_XML*
*CPA_ACC_SVC_TYPE_VIDEO_ANALYTICS* Video
*CPA _ ACC _ SVC _ TYPE _ VIDEO _ ANALYTICS 视频*
Analytics
分析学

## enum _CpaInstanceState

列举型别_CpaInstanceStat

Instance State
实例状态

**Deprecated:**
Deprecated:
As of v1.3 of the Crypto API, this enum has been deprecated, replaced by **CpaOperationalState**.
从 Crypto API 1.3 版开始，此枚举已被取代，由 CpaOperationalState

Enumeration of the different instance states that are possible.
可能的不同实例状态的枚举。

**Enumerator:**
**枚举器:**

*CPA_INSTANCE_STATE_INITIALISED* Instance is in the initialized state and ready for use.
*CPA _ INSTANCE _ STATE _ INITIALISED 实例处于初始化状态，可以使用。*

*CPA_INSTANCE_STATE_SHUTDOWN* Instance is in the shutdown state and not available for
*CPA_INSTANCE_STATE_SHUTDOWN 实例处于关闭状态，不可用于*
use.
使用。

## enum _CpaOperationalState

列举型别_CpaOperationalStat

Instance operational state
实例操作状态

Enumeration of the different operational states that are possible.
列举可能的不同操作状态。

**Enumerator:**
**枚举器:**

*CPA_OPER_STATE_DOWN* Instance is not available for use. May not yet be initialized, or

*CPA_OPER_STATE_DOWN 例程不可用。可能尚未初始化，或者*

stopped.

停了。

*CPA_OPER_STATE_UP* Instance is available for use. Has been initialized and started.

*CPA _ OPER _状态_UP 实例可供使用。已初始化并启动。*

enum **_CpaInstanceEvent**

**列举型别_CpaInstanceEven**

Instance Events

实例事件

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

将导致调用注册实例通知回调函数的不同事件的枚举。

**Enumerator:**

**枚举器:**

*CPA_INSTANCE_EVENT_RESTARTING* Event type that triggers the registered instance

*CPA_INSTANCE_EVENT_RESTARTING 触发注册实例的事件类型*

notification callback function when and instance is restarting. The reason why an instance is restarting is implementation specific. For example a hardware implementation may send this event if the hardware device is about to be reset.

实例重新启动时的通知回调函数。实例重新启动的原因是特定于实现的。例如，如果硬件设备将要被重置，则硬件实现可以发送该事件。

*CPA_INSTANCE_EVENT_RESTARTED* Event type that triggers the registered instance

*触发注册实例的 CPA _ INSTANCE _ EVENT _ RESTARTED 事件类型*

notification callback function when and instance has restarted. The reason why an instance has restarted is implementation specific. For example a hardware implementation may send this event after the hardware device has been reset.

实例重新启动时的通知回调函数。实例重新启动的原因是特定于实现的。例如，硬件实现可以在硬件设备复位后发送该事件。

*CPA_INSTANCE_EVENT_FATAL_ERROR* Event type that triggers the registered instance

*触发注册实例的 CPA_INSTANCE_EVENT_FATAL_ERROR 事件类型*

notification callback function when an error has been detected that requires the device to be reset. This event will be sent by all instances using the device, both on the host and guests.

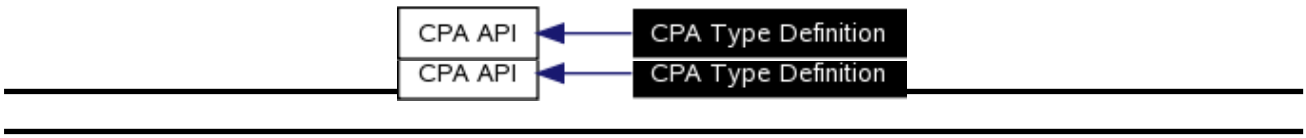
当检测到需要重置设备的错误时的通知回调功能。该事件将由主机和客户机上使用该设备的所有实例发送。

# 4 CPA Type Definition
# 5 CPA 类型定义

**[CPA API]**

[CPA API]

Collaboration diagram for CPA Type Definition:
CPA 类型定义的协作图:




## 5.1 Detailed Description
## 5.2 详细描述

**File: cpa_types.h**

文件:cpa_types.h

This is the CPA Type Definitions.
这是 CPA 类型定义。

## 5.3 Defines
## 5.4 界定

- #define **NULL**

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

## 5.5 Typedefs
## 5.6 类型定义

- typedef uint8_t **Cpa8U**

- typedef uint8_tCpa8U
- typedef int8_t **Cpa8S**
- typedef int8_tCpa8S
- typedef uint16_t **Cpa16U**
- typedef uint16_tCpa16U
- typedef int16_t **Cpa16S**
- typedef int16_tCpa16S
- typedef uint32_t **Cpa32U**
- typedef uint32_tCpa32U
- typedef int32_t **Cpa32S**
- typedef int32_tCpa32S
- typedef uint64_t **Cpa64U**
- typedef uint64_tCpa64U
- typedef int64_t **Cpa64S**
- typedef int64_tCpa64S
- typedef enum **_CpaBoolean CpaBoolean**
- typedef 枚举_CpaBoolean CpaBoolean

## 5.7 Enumerations
## 5.8 列举

- enum **_CpaBoolean** {
  **CPA_FALSE**,
  **CPA_TRUE**

- 列举型别_CpaBoolean
  CPA_FALSECPA_TRUE
  }
  }

---

## 5.9 Define Documentation
## 5.10 定义文档

```
#define NULL
```
#定义 NULL

**File: cpa_types.h**
文件:cpa_types.h

4.5 Define Documentation

NULL definition.

4.6 定义文档空定义。

Declare a bitmap of specified size (in bits).
声明指定大小的位图(以位为单位)。

This macro is used to declare a bitmap of arbitrary size.
这个宏用于声明任意大小的位图。

To test whether a bit in the bitmap is set, use **CPA_BITMAP_BIT_TEST**.
若要测试位图中的某个位是否已设置，请使用 CPA_BITMAP_BIT_TEST

While most uses of bitmaps on the API are read-only, macros are also provided to set (see **CPA_BITMAP_BIT_SET**) and clear (see **CPA_BITMAP_BIT_CLEAR**) bits in the bitmap.
虽然位图在 API 上的大多数使用都是只读的，但是也提供了宏来设置(请参见 CPA_BITMAP_BIT_SET)和清晰(参见 CPA_BITMAP_BIT_CLEAR

Test a specified bit in the specified bitmap. The bitmap may have been declared using **CPA_BITMAP**. Returns a Boolean (true if the bit is set, false otherwise).
测试指定位图中的指定位。位图可能已经使用 CPA_BITMAP。返回一个布尔值(如果该位被置位，则为真，否则为假)。

**File: cpa_types.h**

文件:cpa_types.h

Set a specified bit in the specified bitmap. The bitmap may have been declared using **CPA_BITMAP**.
在指定的位图中设置指定的位。位图可能已经使用 CPA_BITMAP

Clear a specified bit in the specified bitmap. The bitmap may have been declared using **CPA_BITMAP**.
清除指定位图中的指定位。位图可能已经使用 CPA_BITMAP

Declare a function or type and mark it as deprecated so that usages get flagged with a warning.
声明一个函数或类型，并将其标记为已弃用，以便用警告标记使用实例。

---

## 4.7 Typedef Documentation

## 4.8 Typedef 文档

**File: cpa_types.h**

文件:cpa_types.h

Unsigned byte base type.
无符号字节基本类型。

typedef int8_t **Cpa8S**

**File: cpa_types.h**

文件:cpa_types.h

Signed byte base type.
有符号字节基类型。

typedef uint16_t **Cpa16U**

**File: cpa_types.h**

文件:cpa_types.h

Unsigned double-byte base type.
无符号双字节基本类型。

## 4.6 Typedef Documentation

## 4.7 Typedef 文档

typedef int16_t **Cpa16S**

**File: cpa_types.h**

**文件:cpa_types.h**

Signed double-byte base type.
有符号双字节基本类型。

typedef uint32_t **Cpa32U**

**File: cpa_types.h**

**文件:cpa_types.h**

Unsigned quad-byte base type.
无符号四字节基本类型。

typedef int32_t **Cpa32S**

**File: cpa_types.h**

**文件:cpa_types.h**

Signed quad-byte base type.
有符号四字节基本类型。

typedef uint64_t **Cpa64U**

**File: cpa_types.h**

**文件:cpa_types.h**

Unsigned double-quad-byte base type.
无符号双四字节基本类型。

typedef int64_t **Cpa64S**

**File: cpa_types.h**

**文件:cpa_types.h**

Signed double-quad-byte base type.
有符号双四字节基本类型。

typedef enum **CpaBoolean** **CpaBoolean**

Boolean type.
布尔类型。

Functions in this API use this type for Boolean variables that take true or false values.
该 API 中的函数将这种类型用于取值为真或假的布尔变量。

## Enumeration Type Documentation

## 枚举类型文档

enum **CpaBoolean**

Boolean type.
布尔类型。

Functions in this API use this type for Boolean variables that take true or false values.
该 API 中的函数将这种类型用于取值为真或假的布尔变量。

**Enumerator:**
**枚举器:**

*CPA_FALSE* False value
*CPA_FALSE 假值*

*CPA_TRUE* True value
*CPA_TRUE 真值*

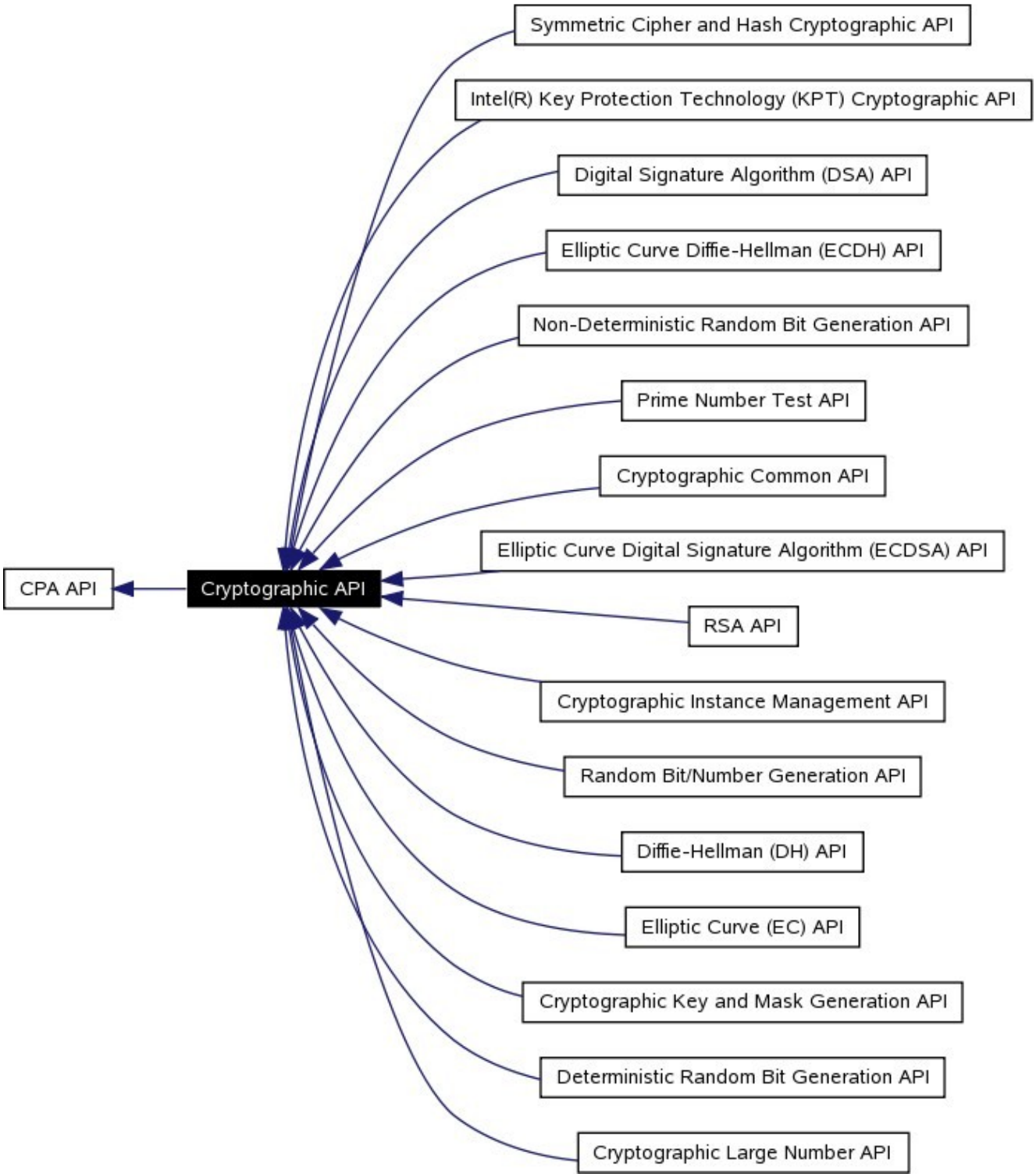
# 6 Cryptographic API

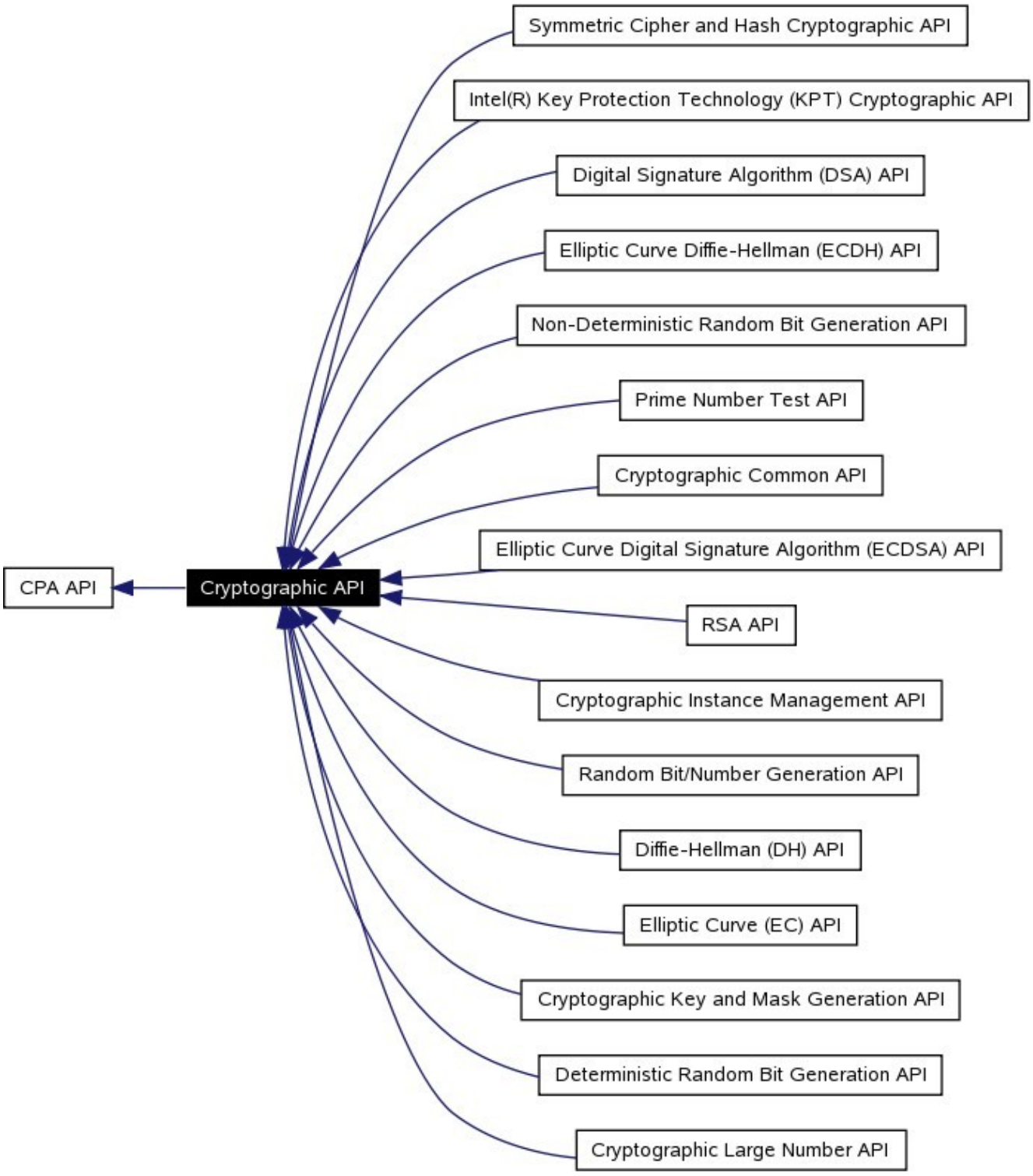
# 7 加密 API

**[CPA API]**

[CPA API]

Collaboration diagram for Cryptographic API:
加密 API 的协作图:




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

7.1 Detailed Description

7.2 详细描述

# 5.1 Detailed Description

# 5.2 详细描述

**File: cpa_cy_common.h**

文件:cpa_cy_common.h

These functions specify the Cryptographic API.
这些函数指定加密 API。

# 5.3 Modules
# 5.4 模块

- **Cryptographic Common API**

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

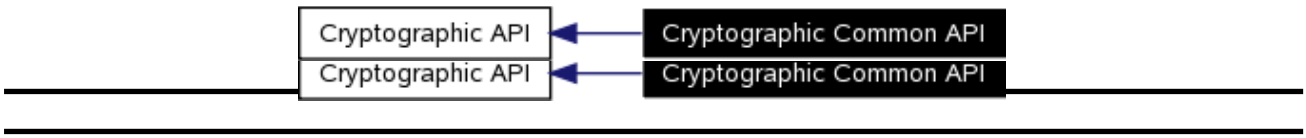
# 8 Cryptographic Common API

# 9 加密公共 API

**[Cryptographic API]**

[Cryptographic API]

Collaboration diagram for Cryptographic Common API:
加密通用 API 的协作图:




## 9.1 Detailed Description

## 9.2 详细描述

**File: cpa_cy_common.h**

文件:cpa_cy_common.h

This file specifies items which are common for both the asymmetric (public key cryptography) and the symmetric operations for the Cryptographic API.
该文件指定了加密 API 的非对称(公钥加密)和对称操作共有的项目。

## 9.3 Typedefs

## 9.4 类型定义

- typedef enum _**CpaCyPriority CpaCyPriority**

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

## 9.5 Enumerations

## 9.6 列举

- enum _**CpaCyPriority** {
  **CPA_CY_PRIORITY_NORMAL**,
  **CPA_CY_PRIORITY_HIGH**

- 列举型别_CpaCyPriority
  CPA_CY_PRIORITY_NORMALCPA_CY_
  PRIORITY_HIGH
  }
  }

## 9.7 Functions

## 9.8 功能

- **CpaStatus cpaCyBufferListGetMetaSize** (const **CpaInstanceHandle** instanceHandle, **Cpa32U**

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

## 9.9 Typedef Documentation

## 9.10 Typedef 文档

typedef enum _**CpaCyPriority CpaCyPriority**

typedef 枚举_CpaCyPrioritCpaCyPriorit

Request priority

请求优先级

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

要给予 API 的请求的优先级的枚举。目前支持两个级别-高和正常。高优先级请求将“尽最大努力”优先于标有正常优先级的请求。

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

Definition of the crypto generic callback function

typedef void(*CpaCyGenericCbFuncCpaStatu

)(void *pCallbackTag,

This data structure specifies the prototype for a generic callback function

该数据结构指定了通用回调函数的原型

**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* Opaque value provided by user while making individual function call.
[in] pCallbackTag 用户在进行单个函数调用时提供的不透明值。

[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 that was submitted in the request
[in]指向请求中提交的操作数据的 pOpData Opaque 指针

**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:**

**cpaCyKeyGenSsl()**
cpaCyKeyGenSsl()

typedef void(* **CpaCyGenFlatBufCbFunc**)(void *pCallbackTag, **CpaStatus** status, void *pOpdata,
typedef void(* CpaCyGenFlatBufCbFunc)(void *pCallbackTag, CpaStatus status, void *pOpdata,

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

使用附加输出 CpaFlatBuffer 参数定义通用回调函数。

This data structure specifies the prototype for a generic callback function which provides an output buffer (of type CpaFlatBuffer).
该数据结构指定了提供输出缓冲区(CpaFlatBuffer 类型)的通用回调函数的原型。

**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* Opaque value provided by user while making individual function call.
[in] pCallbackTag 用户在进行单个函数调用时提供的不透明值。
[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 that was submitted in the request
[in]指向请求中提交的操作数据的 pOpData Opaque 指针
[in] *pOut* Pointer to the output buffer provided in the request invoking this callback.
[in] pOut 指向调用此回调的请求中提供的输出缓冲区的指针。

**Return values:**
**返回值:**

*None*

6.5 Typedef

*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:**

None

没有人

typedef void(* **CpaCyInstanceNotificationCbFunc**)(const **CpaInstanceHandle** instanceHandle, void

Callback function for instance notification support.

实例通知支持的回调函数。

typedef void(*CpaCyInstanceNotificationCbFunc)(const CpaInstanceHandle instanceHandle, void

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

这是实例通知回调函数的原型。回调函数作为参数传递给 cpaCyInstanceSetNotificationCb

**Context:**

**背景:**

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

该功能将在要求不得允许睡眠的环境中执行。

**Assumptions:**

**假设:**

None

没有人

**Side-Effects:**

**副作用:**
None
没有人

**Blocking:**
**阻止:**
No
不

**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 value provided by user while making individual function calls.
[in] pCallbackTag 使用者在进行个别函式呼叫时所提供的不透明值。
[in] *instanceEvent* The event that will trigger this function to get invoked.
[in] instanceEvent 将触发此函数被调用的事件。

**Return values:**
**返回值:**
*None*
*没有人*

**Precondition:**
**前提条件:**
Component has been initialized and the notification function has been set via the cpaCyInstanceSetNotificationCb function.
组件已初始化，并且已通过 cpaCyInstanceSetNotificationCb 函数设置了通知函数。

**Postcondition:**
**后置条件:**
None
没有人

**Note:**

**注意:**

None
没有人

**See also:**

**另请参见:**

**cpaCyInstanceSetNotificationCb()**,
cpaCyInstanceSetNotificationCb(),

---

## 6.6 Enumeration Type Documentation

## 6.7 枚举类型文档

enum CpaCyPriority

Request priority
请求优先级

Enumeration of priority of the request to be given to the API. Currently two levels - HIGH and NORMAL are supported. HIGH priority requests will be prioritized on a "best-effort" basis over requests that are marked with a NORMAL priority.
要给予 API 的请求的优先级的枚举。目前支持两个级别-高和正常。高优先级请求将“尽最大努力”优先于标有正常优先级的请求。

**Enumerator:**
**枚举器:**

*CPA_CY_PRIORITY_NORMAL* Normal priority
*CPA _ CY _优先级_正常正常优先级*
*CPA_CY_PRIORITY_HIGH* High priority
*CPA _ CY _ PRIORITY _高高优先级*

---

## 6.8 Function Documentation

## 6.9 功能文档

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

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

函数返回必须为 CpaBufferList 的 pPrivateMetaData 成员分配的内存大小。

This function is used obtain the size (in bytes) required to allocate a buffer descriptor for the pPrivateMetaData member in the CpaBufferList the structure. Should the function return zero then no meta data is required for the buffer list.
此函数用于获取为 CpaBufferList 结构中的 pPrivateMetaData 成员分配缓冲区描述符所需的大小(以字节为单位)。如果函数返回零，那么缓冲区列表不需要元数据。

**Context:**
**背景:**
This function may be called from any context.
这个函数可以从任何上下文中调用。

**Assumptions:**
**假设:**
None
没有人

**Side-Effects:**
**副作用:**
None
没有人

**Blocking:**
**阻止:**
No
不

**Reentrant:**
**可重入:**
No
不

**Thread-safe:**
**线程安全:**
Yes
是

**Parameters:**
**参数:**
[in] *instanceHandle* Handle to an instance of this API.
[in]此 API 实例的 instanceHandle 句柄。
[in] *numBuffers* The number of pointers in the CpaBufferList. this is the maximum number of CpaFlatBuffers which may be contained in this CpaBufferList.
[in]numBuffers CpaBufferList 中的指针数。这是此 CpaBufferList 中可以包含的 CpaFlatBuffers 的最大数量。
[out] *pSizeInBytes* Pointer to the size in bytes of memory to be allocated when the client

6.7 Function

wishes to allocate a cpaFlatBuffer

[out] pSizeInBytes 当用户端想要配置 cpaFlatBuffer 时，所要配置的记忆体大小(以位元组为单位)指标

**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_UNSUPPORTED* Function is not supported.

*CPA_STATUS_FAIL 函数失败。传递的 CPA_STATUS_INVALID_PARAM 参数无效。不支持 CPA_STATUS_UNSUPPORTED 函数。*

**Precondition:**

**前提条件:**

None.

没有。

**Postcondition:**

**后置条件:**

None

没有人

**Note:**

**注意:**

None

没有人

**See also:**

**另请参见:**

**cpaCyGetInstances()**

cpaCyGetInstances()

**CpaStatus** cpaCyGetStatusText ( const **CpaInstanceHandle** *instanceHandle*,
CpaStatus cpaCyGetStatusText ( const CpaInstanceHandle *instanceHandle*,
**CpaStatus** **CpaInstanceHandle** *instanceHandle*,
**CpaStatus** **Cpa8S** *pStatusText*
Cpa8S *
*pStatusText*

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

函数返回一个字符串，该字符串指示特定实例发生的特定错误。

When a function invocation on a particular instance returns an error, the client can invoke this function to query the instance for a null terminated string which describes the general error condition, and if available additional text on the specific error. The Client MUST allocate CPA_STATUS_MAX_STR_LENGTH_IN_BYTES bytes for the buffer string.
当对特定实例的函数调用返回错误时，客户端可以调用此函数来查询实例，以查找描述一般错误情况的空终止字符串，以及特定错误的附加文本(如果可用)。客户端必须为缓冲区字符串分配 CPA _ STATUS _ MAX _ STR _ LENGTH _ IN _ BYTES 字节。

**Context:**
**背景:**

This function may be called from any context.
这个函数可以从任何上下文中调用。

**Assumptions:**
**假设:**

None
没有人

**Side-Effects:**
**副作用:**

None
没有人

**Blocking:**
**阻止:**

No
不

**Reentrant:**
**可重入:**

No
不

**Thread-safe:**
**线程安全:**

Yes
是

**Parameters:**
**参数:**

[in] *instanceHandle* Handle to an instance of this API.
[in]此 API 实例的 instanceHandle 句柄。
[in] *errStatus* The error condition that occurred
[in] errStatus 发生的错误状况
[out] *pStatusText* Pointer to the string buffer that will be updated with a null terminated

status text string. The invoking application MUST allocate this buffer to be CPA_STATUS_MAX_STR_LENGTH_IN_BYTES.

[out]指向字符串缓冲区的 pStatusText 指针，该缓冲区将使用以空值终止的状态文本字符串进行更新。调用应用程序必须将该缓冲区分配为 CPA _ STATUS _ MAX _ STR _ LENGTH _ IN _ BYTES。

**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. Note, In this scenario it is INVALID to call this function a further time.

*CPA_STATUS_FAIL 函数失败。注意，在这种情况下，调用这个是无效的*进一步发挥作用。

*CPA_STATUS_INVALID_PARAM* Invalid parameter passed in.

*传递的 CPA_STATUS_INVALID_PARAM 参数无效。*

*CPA_STATUS_UNSUPPORTED* Function is not supported.

*不支持 CPA_STATUS_UNSUPPORTED 函数。*

**Precondition:**

**前提条件:**

None.

没有。

**Postcondition:**

**后置条件:**

None

没有人

**Note:**

**注意:**

None

没有人

**See also:**

**另请参见:**

**CpaStatus**

CpaStatus

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

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

CpaStatuCpa16

s cpaCyGetNumInstances(

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

该函数将获取加密 API 的实现所支持的实例数量。然后，该数字用于确定必须传递给的数组的大小 cpaCyGetInstances()

**Context:**
**背景:**

This function MUST NOT be called from an interrupt context as it MAY sleep.
该函数不能从中断上下文中调用，因为它可能会休眠。

**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:**

[out] *pNumInstances* Pointer to where the number of instances will be written.
[out] pNumInstances 指向将写入实例数的位置的指针。

**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_UNSUPPORTED* Function is not supported.

*CPA_STATUS_FAIL 函数失败。传递的 CPA_STATUS_INVALID_PARAM 参数无效。不支持 CPA_STATUS_UNSUPPORTED 函数。*

**Precondition:**

**前提条件:**
None
没有人

**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:**
**cpaCyGetInstances**
cpaCyGetInstances

```
cpaCyGetInstances (                             numInstances
cpaCyGetInstances (                             numInstances,
                              cyInstances
                              *
                            )
```

Get the handles to the instances that are supported by the API implementation.
获取 API 实现所支持的实例的句柄。

This function will return handles to the instances that are supported by an implementation of the Cryptographic API. These instance handles can then be used as input parameters with other Cryptographic API functions.
该函数将返回加密 API 实现所支持的实例的句柄。然后，这些实例句柄可以用作其他加密 API 函数的输入参数。

This function will populate an array that has been allocated by the caller. The size of this API will have been determined by the **cpaCyGetNumInstances()** function.
这个函数将填充一个由调用者分配的数组。这个 API 的大小将由 cpaCyGetNumInstances()

**Context:**

**背景:**

This function MUST NOT be called from an interrupt context as it MAY sleep.
该函数不能从中断上下文中调用，因为它可能会休眠。

**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] *numInstances* Size of the array. If the value is not the same as the number of instances supported, then an error (**CPA_STATUS_INVALID_PARAM**) is returned.
[in] numInstances 阵列的大小。如果该值与实例，则出现错误(CPA_STATUS_INVALID_PARAM

[in,out] *cyInstances* Pointer to where the instance handles will be written.
[in，out] cyInstances 指向将写入实例句柄的位置的指针。

**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_UNSUPPORTED* Function is not supported.

*CPA_STATUS_FAIL 函数失败。传递的 CPA_STATUS_INVALID_PARAM 参数无效。不支持 CPA_STATUS_UNSUPPORTED 函数。*

**Precondition:**

前提条件:

None

没有人

**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:

**cpaCyGetNumInstances**

cpaCyGetNumInstances

Function to get information on a particular instance.

**CpaStatus CPA_DEPRECATED** cpaCyInstanceGetInfo ( const **CpaInstanceHandle**

CpaStatus CPA_DEPRECATED cpaCyInstanceGetInfo( const CpaInstanceHandle instanceHandle,

*instanceHandle*,

struct **_CpaInstanceInfo** * *pInstanceInfo*

struct _CpaInstanceInfo * pInstanceInfo

函数来获取特定实例的信息。

**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

**cpaCyInstanceGetInfo2**.

cpaCyInstanceGetInfo2。

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

该函数将通过 CpaInstanceInfo 结构提供特定于实例的信息。

**Context:**

背景:

This function may be called from any context.

这个函数可以从任何上下文中调用。

**Assumptions:**

**假设：**

None
没有人

**Side-Effects:**
**副作用：**
None
没有人

**Blocking:**
**阻止：**
No
不

**Reentrant:**
**可重入：**
No
不

**Thread-safe:**
**线程安全：**
Yes
是

**Parameters:**
**参数：**

[in] *instanceHandle* Handle to an instance of this API to be initialized.
[in]要初始化的此 API 实例的 instanceHandle 句柄。
[out] *pInstanceInfo* Pointer to the memory location allocated by the client into which the CpaInstanceInfo structure will be written.
[out] pInstanceInfo 指向由客户端分配的内存位置的指针，CpaInstanceInfo 结构将写入该内存位置。

**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_UNSUPPORTED* Function is not supported.
*CPA_STATUS_FAIL 函数失败。传递的 CPA_STATUS_INVALID_PARAM 参数无效。不支持 CPA_STATUS_UNSUPPORTED 函数。*

**Precondition:**

**前提条件：**

The client has retrieved an instanceHandle from successive calls to **cpaCyGetNumInstances** and
客户端已从对的连续调用中检索到 instanceHandle**cpaCyGetNumInstances**
**cpaCyGetInstances**.

cpaCyGetInstances。

**Postcondition:**

**后置条件:**

None

没有人

**Note:**

**注意:**

None

没有人

**See also:**

**另请参见:**

**cpaCyGetNumInstances**, **cpaCyGetInstances**, CpaInstanceInfo

cpaCyGetNumInstances, cpaCyGetInstances

Function to get information on a particular instance.

函数来获取特定实例的信息。

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

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

该函数将通过一个 CpaInstanceInfo2

**cpaCyInstanceGetInfo**.

cpaCyInstanceGetInfo。

**Context:**

**背景:**

This function may be called from any context.

这个函数可以从任何上下文中调用。

**Assumptions:**

**假设:**

None

没有人

**Side-Effects:**

**副作用:**

None

没有人

**Blocking:**

**阻止:**

No
不

**Reentrant:**
**可重入:**

No
不

**Thread-safe:**
**线程安全:**

Yes
是

**Parameters:**
**参数:**

[in] *instanceHandle* Handle to an instance of this API to be initialized.
[in]要初始化的此 API 实例的 instanceHandle 句柄。
[out] *pInstanceInfo2* Pointer to the memory location allocated by the client into which the CpaInstanceInfo2 structure will be written.
[out] pInstanceInfo2 指向客户端分配的内存位置的指针，CpaInstanceInfo2 结构将写入该内存位置。

**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_UNSUPPORTED* Function is not supported.
*CPA_STATUS_FAIL 函数失败。传递的 CPA_STATUS_INVALID_PARAM 参数无效。不支持 CPA_STATUS_UNSUPPORTED 函数。*

**Precondition:**

**前提条件:**

The client has retrieved an instanceHandle from successive calls to **cpaCyGetNumInstances** and
客户端已从对的连续调用中检索到 instanceHandle**cpaCyGetNumInstances**
**cpaCyGetInstances**.
cpaCyGetInstances。

**Postcondition:**
**后置条件:**

None
没有人

**Note:**

**注意:**

None
没有人

**See also:**

另请参见:

**cpaCyGetNumInstances**, **cpaCyGetInstances**, CpaInstanceInfo
cpaCyGetNumInstances, cpaCyGetInstances

**CpaStatus**
CpaStatus
cpaCyInstanceSetNotificationCb ( const **CpaInstanceHandle**
cpaCyInstanceSetNotificationCb(const CpaInstanceHandle
const
常数
**CpaCyInstanceNotificationCbFunc**
CpaCyInstanceNotificationCbFunc

| | | |
|---|---|---|
| *instanceHandle*, *pInstanceNotificationCb*, | void * | *实例句柄，pInstanceNotificationCb，pCallbackTag* |

Subscribe for instance notifications.
订阅实例通知。

6.7 Function

```
  void * pCallbackTag                          )
)
```

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

CpaCy 接口的客户端可以通过注册一个
**CpaCyInstanceNotificationCbFunc** function.
CpaCyInstanceNotificationCbFunc 功能。

**Context:**
**背景:**
This function may be called from any context.
这个函数可以从任何上下文中调用。

**Assumptions:**
**假设:**
None
没有人

**Side-Effects:**
**副作用:**
None
没有人

**Blocking:**
**阻止:**
No
不

**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] *pInstanceNotificationCb* Instance notification callback function pointer.
[in] pInstanceNotificationCb 实例通知回调函数指针。
[in] *pCallbackTag* Opaque value provided by user while making individual function calls.
[in] pCallbackTag 不透明值由用户在执行单个函数时提供电话。

**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_UNSUPPORTED* Function is not supported.
*CPA_STATUS_FAIL 函数失败。传递的 CPA_STATUS_INVALID_PARAM 参数无效。不支持 CPA_STATUS_UNSUPPORTED 函数。*

**Precondition:**

**前提条件:**

Instance has been initialized.
实例已初始化。

**Postcondition:**
**后置条件:**

None
没有人

**Note:**

**注意:**

None
没有人

**See also:**

**另请参见:**

**CpaCyInstanceNotificationCbFunc**

CpaCyInstanceNotificationCbFunc

# 10 Cryptographic Instance Management API
# 11 加密实例管理 API

**[Cryptographic API]**

[Cryptographic API]

Collaboration diagram for Cryptographic Instance Management API:
加密实例管理 API 的协作图:


## 11.1 Detailed Description
## 11.2 详细描述

**File: cpa_cy_im.h**

文件:cpa_cy_im.h

These functions specify the Instance Management API for available Cryptographic Instances. It is expected that these functions will only be called via a single system maintenance entity, rather than individual clients.
这些函数为可用的加密实例指定了实例管理 API。预计这些功能将仅通过单个系统维护实体调用，而不是单独的客户端。

## 11.3 Data Structures
## 11.4 数据结构

- struct **_CpaCyCapabilitiesInfo**
- 结构体_CpaCyCapabilitiesInfo

## 11.5 Typedefs
## 11.6 类型定义

- typedef **_CpaCyCapabilitiesInfo CpaCyCapabilitiesInfo**
- 数据类型说明_CpaCyCapabilitiesInfo CpaCyCapabilitiesInfo

## 11.7 Functions
## 11.8 功能

- **CpaStatus cpaCyStartInstance** (**CpaInstanceHandle** instanceHandle)

- CpaStatus cpaCyStartInstance （CpaInstanceHandle
- **CpaStatus cpaCyStopInstance** (**CpaInstanceHandle** instanceHandle)
- CpaStatus cpaCyStopInstance （CpaInstanceHandle
- **CpaStatus cpaCyQueryCapabilities** (const **CpaInstanceHandle** instanceHandle,
- CpaStatus cpaCyQueryCapabilities（常量 CpaInstanceHandle
  **CpaCyCapabilitiesInfo** *pCapInfo)
  CpaCyCapabilitiesInfo * pCapInfo)
- **CpaStatus cpaCySetAddressTranslation** (const **CpaInstanceHandle** instanceHandle,
- CpaStatus cpaCySetAddressTranslation（常量 CpaInstanceHandle
  **CpaVirtualToPhysical** virtual2Physical)
  CpaVirtualToPhysical 虚拟 2 物理）

---

## 11.9 Data Structure Documentation
## 11.10 数据结构文档

### 7.5.1 _CpaCyCapabilitiesInfo Struct Reference

### 7.5.2 _CpaCyCapabilitiesInfo 结构引用

#### 7.5.2.1 Detailed Description
#### 7.5.2.2 详细描述

Cryptographic Capabilities Info
加密功能信息

This structure contains the capabilities that vary across API implementations. This structure is used in conjunction with **cpaCyQueryCapabilities()** to determine the capabilities supported by a particular API implementation.
这个结构包含不同 API 实现的功能。此结构与一起使用 cpaCyQueryCapabilities()

The client MUST allocate memory for this structure and any members that require memory. When the structure is passed into the function ownership of the memory passes to the function. Ownership of the memory returns to the client when the function returns.
客户端必须为此结构和任何需要内存的成员分配内存。当结构被传递给函数时，内存的所有权就传递给了函数。当函数返回时，内存的所有权返回给客户端。

7.5.1 _CpaCyCapabilitiesInfo Struct Reference

7.5.1 _CpaCyCapabilitiesInfo 结构引用

**7.5.2.3 Data Fields**

**7.5.2.4 数据字段**

- **CpaBoolean symSupported**
- CpaBoolean symSupported
- **CpaBoolean symDpSupported**
- CpaBoolean symDpSupported
- **CpaBoolean dhSupported**
- CpaBoolean dhSupported
- **CpaBoolean dsaSupported**
- CpaBoolean dsaSupported
- **CpaBoolean rsaSupported**
- CpaBoolean rsaSupported
- **CpaBoolean ecSupported**
- CpaBoolean ecSupported
- **CpaBoolean ecdhSupported**
- CpaBoolean ecdhSupported
- **CpaBoolean ecdsaSupported**
- CpaBoolean ecdsaSupported
- **CpaBoolean keySupported**
- CpaBoolean keySupported
- **CpaBoolean lnSupported**
- CpaBoolean lnSupported
- **CpaBoolean primeSupported**
- CpaBoolean primeSupported
- **CpaBoolean drbgSupported**
- CpaBoolean drbgSupported
- **CpaBoolean nrbgSupported**
- CpaBoolean nrbgSupported
- **CpaBoolean randSupported**
- CpaBoolean randSupported
- **CpaBoolean kptSupported**
- CpaBoolean kptSupported
- **CpaBoolean hkdfSupported**
- CpaBoolean hkdfSupported
- **CpaBoolean extAlgchainSupported**
- CpaBoolean extAlgchainSupported
- **CpaBoolean ecEdMontSupported**
- CpaBoolean ecEdMontSupported
- **CpaBoolean ecSm2Supported**
- CpaBoolean ecSm2Supported

**7.5.2.5 Field Documentation**
**7.5.2.6 现场文件**

**CpaBoolean CpaCyCapabilitiesInfo::symSupported**

CPA_TRUE if instance supports the symmetric cryptography API. See **Symmetric Cipher and Hash Cryptographic API**.

CPA_TRUE，如果实例支持对称加密 API。看见 Symmetric Cipher and HashCryptographic API

**CpaBoolean CpaCyCapabilitiesInfo::symDpSupported**

CPA_TRUE if instance supports the symmetric cryptography data plane API. See **Symmetric cryptographic Data Plane API**.
CPA_TRUE，如果实例支持对称加密数据平面 API。看见 Symmetriccryptographic Data Plane API

CPA_TRUE if instance supports the Diffie Hellman API. See **Diffie-Hellman (DH) API**.
CPA_TRUE 如果实例支持 Diffie Hellman API。看见 Diffie-Hellman (DH) API

CPA_TRUE if instance supports the DSA API. See **Digital Signature Algorithm (DSA) API**.
CPA_TRUE 如果实例支持 DSA API。看见 Digital Signature Algorithm (DSA) API

CPA_TRUE if instance supports the RSA API. See **RSA API**.
CPA_TRUE 如果实例支持 RSA API。看见 RSA API

CPA_TRUE if instance supports the Elliptic Curve API. See **Elliptic Curve (EC) API**.
CPA_TRUE 如果实例支持椭圆曲线 API。看见 Elliptic Curve (EC) API

CPA_TRUE if instance supports the Elliptic Curve Diffie Hellman API. See **Elliptic Curve Diffie-Hellman (ECDH) API**.
CPA_TRUE 如果实例支持椭圆曲线 Diffie Hellman API。看见 Elliptic Curve Diffie-Hellman(ECDH) API

CPA_TRUE if instance supports the Elliptic Curve DSA API. See **Elliptic Curve Digital Signature Algorithm (ECDSA) API**.
CPA_TRUE 如果实例支持椭圆曲线 DSA API。看见 Elliptic Curve Digital SignatureAlgorithm (ECDSA) API

CPA_TRUE if instance supports the Key Generation API. See **Cryptographic Key and Mask Generation API**.
CPA_TRUE 如果实例支持密钥生成 API。看见 Cryptographic Key and Mask GenerationAPI

CPA_TRUE if instance supports the Large Number API. See **Cryptographic Large Number API**.
CPA_TRUE 如果实例支持大数 API。看见 Cryptographic Large Number API

CPA_TRUE if instance supports the prime number testing API. See **Prime Number Test API**.
如果实例支持素数测试 API，则为 CPA_TRUE。看见 Prime Number Test API

CPA_TRUE if instance supports the DRBG API. See **Deterministic Random Bit Generation API**.
CPA_TRUE 如果实例支持 DRBG API。看见 Deterministic Random Bit GenerationAPI

CPA_TRUE if instance supports the NRBG API. See **Non-Deterministic Random Bit Generation API**.
CPA_TRUE 如果实例支持 NRBG API。看见 Non-Deterministic Random Bit Generation API

CPA_TRUE if instance supports the random bit/number generation API. See **Random Bit/Number Generation API**.
CPA_TRUE 如果实例支持随机位/数生成 API。看见 Random Bit/NumberGeneration API

CPA_TRUE if instance supports the Intel(R) KPT Cryptographic API. See **Intel(R) Key Protection Technology (KPT) Cryptographic API**.
CPA_TRUE 如果实例支持英特尔 KPT 加密 API。看见 Intel(R) Key ProtectionTechnology (KPT) Cryptographic API

CPA_TRUE if instance supports the HKDF components of the KeyGen API. See **Cryptographic Key and Mask Generation API**.
如果实例支持 KeyGen API 的 HKDF 组件，则为 CPA_TRUE。看见 Cryptographic Key andMask Generation API

CPA_TRUE if instance supports algorithm chaining for certain wireless algorithms. Please refer to implementation for details. See **Symmetric Cipher and Hash Cryptographic API**.
如果实例支持某些无线算法的算法链，则为 CPA_TRUE。详情请参考实施。看见 Symmetric Cipher and Hash Cryptographic API

CPA_TRUE if instance supports the Edwards and Montgomery elliptic curves of the EC API. See **Elliptic Curve (EC) API**
CPA_TRUE 如果实例支持 EC API 的 Edwards 和 Montgomery 椭圆曲线。看见 EllipticCurve (EC) API

CPA_TRUE if instance supports the EcSM2 API. See cpaCyEcsm2.
如果实例支持 EcSM2 API，则 CPA_TRUE。参见 cpaCyEcsm2。

## 7.6 Typedef Documentation

## 7.7 Typedef 文档

typedef struct **CpaCyCapabilitiesInfo CpaCyCapabilitiesInfo**

Cryptographic Capabilities Info
加密功能信息

This structure contains the capabilities that vary across API implementations. This structure is used in conjunction with **cpaCyQueryCapabilities()** to determine the capabilities supported by a particular API implementation.
这个结构包含不同 API 实现的功能。此结构与一起使用 cpaCyQueryCapabilities()

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

客户端必须为此结构和任何需要内存的成员分配内存。当结构被传递给函数时，内存的所有权就传递给了函数。当函数返回时，内存的所有权返回给客户端。

---

## 7.8 Function Documentation

## 7.9 功能文档

**CpaStatus** cpaCyStartInstance ( **CpaInstanceHandle** *instanceHandle* )

Cryptographic Component Initialization and Start function.

CpaStatuCpaInstanceHandl

的 cpaCyStartInstance(

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

该函数将初始化并启动加密组件。它必须在调用任何其他加密函数之前被调用。对于每个实例，该函数只应调用一次(无论是第一次调用，还是在成功调用 cpaCyStopInstance 之后)。后续调用将不起作用。

**Context:**
**背景:**

This function may sleep, and MUST NOT be called in interrupt context.
该函数可能会休眠，不得在中断上下文中调用。

**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:**
[out] *instanceHandle* Handle to an instance of this API to be initialized.
[out]要初始化的此 API 实例的 instanceHandle 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. Suggested course of action is to shutdown and
*CPA_STATUS_FAIL 函数失败。建议的行动方案是关闭和*
restart.
重启。
*CPA_STATUS_UNSUPPORTED* Function is not supported.
*不支持 CPA_STATUS_UNSUPPORTED 函数。*

**Precondition:**
**前提条件:**
None.
没有。

**Postcondition:**
**后置条件:**
None
没有人

**Note:**
**注意:**
Note that this is a synchronous function and has no completion callback associated with it.
请注意，这是一个同步函数，没有与之关联的完成回调。

**See also:**
**另请参见:**
**cpaCyStopInstance()**
cpaCyStopInstance()

**CpaStatus** cpaCyStopInstance ( **CpaInstanceHandle** *instanceHandle* )
CpaStatuCpaInstanceHandl

Cryptographic Component Stop function.

加密组件停止功能。

This function will stop the Cryptographic component and free all system resources associated with it. The client MUST ensure that all outstanding operations have completed before calling this function. The recommended approach to ensure this is to deregister all session or callback handles before calling this function. If outstanding operations still exist when this function is invoked, the callback function for each of those operations will NOT be invoked and the shutdown will continue. If the component is to be restarted, then a call to cpaCyStartInstance is required.
此函数将停止加密组件并释放与之相关的所有系统资源。在调用这个函数之前，客户端必须确保所有未完成的操作都已经完成。确保这一点的推荐方法是在调用该函数之前注销所有会话或回调句柄。如果调用此函数时仍有未完成的操作，则不会调用这些操作的回调函数，关机将继续。如果要重新启动组件，那么需要调用 cpaCyStartInstance。

**Context:**
**背景：**
This function may sleep, and so MUST NOT be called in interrupt context.
该函数可能会休眠，因此不能在中断上下文中调用。

**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* Handle to an instance of this API to be shutdown.
[in]要关闭的此 API 实例的 instanceHandle 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. Suggested course of action is to ensure
*CPA_STATUS_FAIL 函数失败。建议的行动方针是确保*
requests are not still being submitted and that all sessions are deregistered. If this does not help, then forcefully remove the component from the system.
请求不再提交，所有会话都已取消注册。如果这不起作用，请从系统中强行卸下组件。

*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.
该组件已通过 cpaCyStartInstance 初始化。

**Postcondition:**
**后置条件:**
None
没有人

**Note:**

**注意:**
Note that this is a synchronous function and has no completion callback associated with it.
请注意，这是一个同步函数，没有与之关联的完成回调。

**See also:**

**另请参见:**
**cpaCyStartInstance()**
cpaCyStartInstance()

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

Returns capabilities of a Cryptographic API instance

This function is used to query the instance capabilities.

返回加密 API 实例的功能。此函数用于查询实例功能。

**Context:**
**背景:**

The function shall not be called in an interrupt context.
该函数不应在中断上下文中调用。

**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* Handle to an instance of this API.
[in]此 API 实例的 instanceHandle 句柄。
[out] *pCapInfo* Pointer to capabilities info structure. All fields in the structure are populated by the API instance.
[out]指向功能信息结构的 pCapInfo 指针。结构中的所有字段都由 API 实例填充。

**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.

7.7 Function

*CPA_STATUS_INVALID_PARAM* Invalid parameter passed in.

*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。不支持 CPA_STATUS_UNSUPPORTED 函数。*

**Precondition:**

**前提条件:**

The instance has been initialized via the **cpaCyStartInstance** function.

实例已通过初始化 cpaCyStartInstance

**Postcondition:**

**后置条件:**

None

没有人

Sets the address translation function

设置地址转换功能

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

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

该函数用于设置实例的虚拟到物理地址转换例程。实例使用指定的例程来执行任何所需的虚拟地址到物理地址的转换。如果应用程序不调用这个函数，那么实例将使用它的默认方法，比如 virt2phys，进行地址转换。

**Context:**

**背景:**

The function shall not be called in an interrupt context.

该函数不应在中断上下文中调用。

**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* Handle to an instance of this API.
[in]此 API 实例的 instanceHandle 句柄。
[in] *virtual2Physical* Routine that performs virtual to physical address translation.
[in]virtual 2 执行虚拟地址到物理地址转换的物理例程。

**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_UNSUPPORTED* Function is not supported.
*CPA_STATUS_FAIL 函数失败。传递的 CPA_STATUS_INVALID_PARAM 参数无效。不支持 CPA_STATUS_UNSUPPORTED 函数。*

**Precondition:**

**前提条件:**
None
没有人

**Postcondition:**
**后置条件:**
None
没有人

**See also:**

**另请参见:**

None

没有人

# 12 Symmetric Cipher and Hash Cryptographic API

# 13 对称密码和散列加密 API

**[Cryptographic API]**

[Cryptographic API]

Collaboration diagram for Symmetric Cipher and Hash Cryptographic API:
对称密码和哈希加密 API 的协作图:


## 13.1 Detailed Description

## 13.2 详细描述

**File: cpa_cy_sym.h**

文件:cpa_cy_sym.h

These functions specify the Cryptographic API for symmetric cipher, hash, and combined cipher and hash operations.
这些函数为对称密码、哈希以及组合密码和哈希操作指定了加密 API。

## 13.3 Modules

## 13.4 模块

- **Symmetric cryptographic Data Plane API**
- Symmetric cryptographic Data Plane API

## 13.5 Data Structures

## 13.6 数据结构

- struct **_CpaCySymCipherSetupData**
- 结构体_CpaCySymCipherSetupData
- struct **_CpaCySymHashNestedModeSetupData**
- 结构体_CpaCySymHashNestedModeSetupData

- struct _**CpaCySymHashAuthModeSetupData**
- 结构体_CpaCySymHashAuthModeSetupData
- struct _**CpaCySymHashSetupData**
- 结构体_CpaCySymHashSetupData
- struct _**CpaCySymSessionSetupData**
- 结构体_CpaCySymSessionSetupData
- struct _**CpaCySymSessionUpdateData**
- 结构体_CpaCySymSessionUpdateData
- struct _**CpaCySymOpData**
- 结构体_CpaCySymOpData
- struct _**CpaCySymStats**
- 结构体_CpaCySymStats
- struct _**CpaCySymStats64**
- 结构体_CpaCySymStats64
- struct _**CpaCySymCapabilitiesInfo**
- 结构体_CpaCySymCapabilitiesInfo

## 13.7 Defines
## 13.8 界定

- #define **CPA_CY_SYM_CIPHER_CAP_BITMAP_SIZE**

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

## 13.9 Typedefs
## 13.10 类型定义

- typedef void * **CpaCySymSessionCtx**

- typedef void *CpaCySymSessionCtx
- typedef enum _**CpaCySymPacketType CpaCySymPacketType**
- typedef 枚举_CpaCySymPacketType CpaCySymPacketType
- typedef enum _**CpaCySymOp CpaCySymOp**
- typedef 枚举_CpaCySymOp CpaCySymOp
- typedef enum _**CpaCySymCipherAlgorithm CpaCySymCipherAlgorithm**
- typedef 枚举_CpaCySymCipherAlgorithm CpaCySymCipherAlgorithm
- typedef enum _**CpaCySymCipherDirection CpaCySymCipherDirection**
- typedef 枚举_CpaCySymCipherDirection CpaCySymCipherDirection
- typedef _**CpaCySymCipherSetupData CpaCySymCipherSetupData**
- 数据类型说明_CpaCySymCipherSetupData CpaCySymCipherSetupData
- typedef enum _**CpaCySymHashMode CpaCySymHashMode**
- typedef 枚举_CpaCySymHashMode CpaCySymHashMode
- typedef enum _**CpaCySymHashAlgorithm CpaCySymHashAlgorithm**
- typedef 枚举_CpaCySymHashAlgorithm CpaCySymHashAlgorithm
- typedef _**CpaCySymHashNestedModeSetupData CpaCySymHashNestedModeSetupData**
- 数据类型说明_CpaCySymHashNestedModeSetupData CpaCySymHashNestedModeSetupData

8.5 Typedefs

8.6类型定义

- typedef _**CpaCySymHashAuthModeSetupData CpaCySymHashAuthModeSetupData**
- 数据类型说明_CpaCySymHashAuthModeSetupData CpaCySymHashAuthModeSetupData
- typedef _**CpaCySymHashSetupData CpaCySymHashSetupData**
- 数据类型说明_CpaCySymHashSetupData CpaCySymHashSetupData
- typedef enum _**CpaCySymAlgChainOrder CpaCySymAlgChainOrder**
- typedef 枚举_CpaCySymAlgChainOrder CpaCySymAlgChainOrder
- typedef _**CpaCySymSessionSetupData CpaCySymSessionSetupData**
- 数据类型说明_CpaCySymSessionSetupData CpaCySymSessionSetupData
- typedef _**CpaCySymSessionUpdateData CpaCySymSessionUpdateData**
- 数据类型说明_CpaCySymSessionUpdateData CpaCySymSessionUpdateData
- typedef _**CpaCySymOpData CpaCySymOpData**
- 数据类型说明_CpaCySymOpData CpaCySymOpData
- typedef _**CpaCySymStats CPA_DEPRECATED**
- 数据类型说明_CpaCySymStats CPA_DEPRECATED
- typedef _**CpaCySymStats64 CpaCySymStats64**
- 数据类型说明_CpaCySymStats64 CpaCySymStats64
- typedef void(* **CpaCySymCbFunc** )(void *pCallbackTag, **CpaStatus** status, const **CpaCySymOp**
- typedef void(*CpaCySymCbFunc CpaStatus CpaCySymOp operationType, void *pOpData, **CpaBufferList** *pDstBuffer, **CpaBoolean** verifyResult) operationType, void *pOpData, CpaBufferList CpaBoolean
- typedef _**CpaCySymCapabilitiesInfo CpaCySymCapabilitiesInfo**
- 数据类型说明_CpaCySymCapabilitiesInfo CpaCySymCapabilitiesInfo

## 8.7 Enumerations

## 8.8 列举

- enum _**CpaCySymPacketType** {
  **CPA_CY_SYM_PACKET_TYPE_FULL**,
  **CPA_CY_SYM_PACKET_TYPE_PARTIAL**,
  **CPA_CY_SYM_PACKET_TYPE_LAST_PARTIAL**

- 列举型别_CpaCySymPacketType
  CPA_CY_SYM_PACKET_TYPE_FULLCPA_CY_SYM_PACKET_
  TYPE_PARTIALCPA_CY_SYM_PACKET_TYPE_LAST_PARTI
  AL
  }
  }
- enum _**CpaCySymOp** {
  **CPA_CY_SYM_OP_NONE**,
  **CPA_CY_SYM_OP_CIPHER**,
  **CPA_CY_SYM_OP_HASH**,
  **CPA_CY_SYM_OP_ALGORITHM_CHAINING**
- 列举型别_CpaCySymOp
  CPA_CY_SYM_OP_NONECPA_CY_SYM_OP_CIPHERCPA
  _CY_SYM_OP_HASHCPA_CY_SYM_OP_ALGORITHM_CH
  AINING
  }
  }
- enum _**CpaCySymCipherAlgorithm** {
  **CPA_CY_SYM_CIPHER_NULL**,
  **CPA_CY_SYM_CIPHER_ARC4**,
  **CPA_CY_SYM_CIPHER_AES_ECB**,
  **CPA_CY_SYM_CIPHER_AES_CBC**,
  **CPA_CY_SYM_CIPHER_AES_CTR**,

**CPA_CY_SYM_CIPHER_AES_CCM**,
**CPA_CY_SYM_CIPHER_AES_GCM**,
**CPA_CY_SYM_CIPHER_DES_ECB**,
**CPA_CY_SYM_CIPHER_DES_CBC**,
**CPA_CY_SYM_CIPHER_3DES_ECB**,
**CPA_CY_SYM_CIPHER_3DES_CBC**,
**CPA_CY_SYM_CIPHER_3DES_CTR**,
**CPA_CY_SYM_CIPHER_KASUMI_F8**,
**CPA_CY_SYM_CIPHER_SNOW3G_UEA2**,
**CPA_CY_SYM_CIPHER_AES_F8**,
**CPA_CY_SYM_CIPHER_AES_XTS**,
**CPA_CY_SYM_CIPHER_ZUC_EEA3**,
**CPA_CY_SYM_CIPHER_CHACHA**,
**CPA_CY_SYM_CIPHER_SM4_ECB**,
**CPA_CY_SYM_CIPHER_SM4_CBC**,
**CPA_CY_SYM_CIPHER_SM4_CTR**

- 列举型别_CpaCySymCipherAlgorithm
  CPA_CY_SYM_CIPHER_NULLCPA_CY_SYM_CIPHER
  _ARC4CPA_CY_SYM_CIPHER_AES_ECBCPA_CY_SY
  M_CIPHER_AES_CBCCPA_CY_SYM_CIPHER_AES_C
  TRCPA_CY_SYM_CIPHER_AES_CCMCPA_CY_SYM_C
  IPHER_AES_GCMCPA_CY_SYM_CIPHER_DES_ECBC
  PA_CY_SYM_CIPHER_DES_CBCCPA_CY_SYM_CIPH
  ER_3DES_ECBCPA_CY_SYM_CIPHER_3DES_CBCCP
  A_CY_SYM_CIPHER_3DES_CTRCPA_CY_SYM_CIPH
  ER_KASUMI_F8CPA_CY_SYM_CIPHER_SNOW3G_UE
  A2CPA_CY_SYM_CIPHER_AES_F8CPA_CY_SYM_CI
  PHER_AES_XTSCPA_CY_SYM_CIPHER_ZUC_EEA3C
  PA_CY_SYM_CIPHER_CHACHACPA_CY_SYM_CIPHE
  R_SM4_ECBCPA_CY_SYM_CIPHER_SM4_CBCCPA_C
  Y_SM4_CIPHER_SM4_CTR

}
}

- enum **_CpaCySymCipherDirection** {
  **CPA_CY_SYM_CIPHER_DIRECTION_ENCRYPT**,
  **CPA_CY_SYM_CIPHER_DIRECTION_DECRYPT**
- 列举型别_CpaCySymCipherDirection
  CPA_CY_SYM_CIPHER_DIRECTION_ENCRYPTCPA_CY_SYM
  _CIPHER_DIRECTION_DECRYPT

}
}

- enum **_CpaCySymHashMode** {
  **CPA_CY_SYM_HASH_MODE_PLAIN**,
  **CPA_CY_SYM_HASH_MODE_AUTH**,
  **CPA_CY_SYM_HASH_MODE_NESTED**
- 列举型别_CpaCySymHashMode
  CPA_CY_SYM_HASH_MODE_PLAINCPA_CY_SYM
  _HASH_MODE_AUTHCPA_CY_SYM_HASH_MODE_
  NESTED

**}**

}

- enum **_CpaCySymHashAlgorithm** {
  **CPA_CY_SYM_HASH_NONE**,
  **CPA_CY_SYM_HASH_MD5**,
  **CPA_CY_SYM_HASH_SHA1**,
  **CPA_CY_SYM_HASH_SHA224**,
  **CPA_CY_SYM_HASH_SHA256**,
  **CPA_CY_SYM_HASH_SHA384**,
  **CPA_CY_SYM_HASH_SHA512**,
  **CPA_CY_SYM_HASH_AES_XCBC**,
  **CPA_CY_SYM_HASH_AES_CCM**,
  **CPA_CY_SYM_HASH_AES_GCM**,
  **CPA_CY_SYM_HASH_KASUMI_F9**,
  **CPA_CY_SYM_HASH_SNOW3G_UIA2**,
  **CPA_CY_SYM_HASH_AES_CMAC**,
  **CPA_CY_SYM_HASH_AES_GMAC**,
  **CPA_CY_SYM_HASH_AES_CBC_MAC**,
  **CPA_CY_SYM_HASH_ZUC_EIA3**,
  **CPA_CY_SYM_HASH_SHA3_256**,
  **CPA_CY_SYM_HASH_SHA3_224**,
  **CPA_CY_SYM_HASH_SHA3_384**,
  **CPA_CY_SYM_HASH_SHA3_512**,
  **CPA_CY_SYM_HASH_SHAKE_128**,
  **CPA_CY_SYM_HASH_SHAKE_256**,
  **CPA_CY_SYM_HASH_POLY**,
  **CPA_CY_SYM_HASH_SM3**
- 列举型别_CpaCySymHashAlgorithm
  CPA_CY_SYM_HASH_NONECPA_CY_SYM_HASH_M
  D5CPA_CY_SYM_HASH_SHA1CPA_CY_SYM_HASH
  _SHA224CPA_CY_SYM_HASH_SHA256CPA_CY_S
  YM_HASH_SHA384CPA_CY_SYM_HASH_SHA512C
  PA_CY_SYM_HASH_AES_XCBCCPA_CY_SYM_HAS
  H_AES_CCMCPA_CY_SYM_HASH_AES_GCMCPA_C
  Y_SYM_HASH_KASUMI_F9CPA_CY_SYM_HASH_S
  NOW3G_UIA2CPA_CY_SYM_HASH_AES_CMACCPA
  _CY_SYM_HASH_AES_GMACCPA_CY_SYM_HASH_
  AES_CBC_MACCPA_CY_SYM_HASH_ZUC_EIA3CP
  A_CY_SYM_HASH_SHA3_256CPA_CY_SYM_HASH
  _SHA3_224CPA_CY_SYM_HASH_SHA3_384CPA_
  CY_SYM_HASH_SHA3_512CPA_CY_SYM_HASH_S
  HAKE_128CPA_CY_SYM_HASH_SHAKE_256CPA_
  CY_SYM_HASH_POLYCPA_CY_SYM_HASH_SM3

**}**

}

- enum **_CpaCySymAlgChainOrder** {
  **CPA_CY_SYM_ALG_CHAIN_ORDER_HASH_THEN_CIPHER**,
  **CPA_CY_SYM_ALG_CHAIN_ORDER_CIPHER_THEN_HASH**
- 列举型别_CpaCySymAlgChainOrder
  CPA_CY_SYM_ALG_CHAIN_ORDER_HASH_THEN_CIPHERCPA_CY_SYM_ALG
  _CHAIN_ORDER_CIPHER_THEN_HASH

**}**

}

## 8.8 Functions
## 8.9 功能

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

# 8.9 Data Structure Documentation

# 8.10 数据结构文档

## 8.10.1 _CpaCySymCipherSetupData Struct Reference

## 8.10.2 _CpaCySymCipherSetupData 结构引用

### 8.10.2.1 Detailed Description
### 8.10.2.2 详细描述

Symmetric Cipher Setup Data.
对称密码设置数据。

This structure contains data relating to Cipher (Encryption and Decryption) to set up a session.
该结构包含与建立会话的密码(加密和解密)相关的数据。

### 8.10.2.3 Data Fields
### 8.10.2.4 数据字段

- **CpaCySymCipherAlgorithm cipherAlgorithm**
- CpaCySymCipherAlgorithm cipherAlgorithm
- **Cpa32U cipherKeyLenInBytes**
- Cpa32U cipherKeyLenInBytes
- **Cpa8U * pCipherKey**
- Cpa8U *pCipherKey
- **CpaCySymCipherDirection cipherDirection**
- CpaCySymCipherDirection cipherDirection

### 8.10.2.5 Field Documentation
### 8.10.2.6 现场文件

**CpaCySymCipherAlgorithm CpaCySymCipherSetupData::cipherAlgorithm**

Cipher algorithm and mode
密码算法和模式

**Cpa32U CpaCySymCipherSetupData::cipherKeyLenInBytes**

Cipher key length in bytes. For AES it can be 128 bits (16 bytes), 192 bits (24 bytes) or 256 bits (32 bytes). For the CCM mode of operation, the only supported key length is 128 bits (16 bytes). For the CPA_CY_SYM_CIPHER_AES_F8 mode of operation, cipherKeyLenInBytes should be set to the combined length of the encryption key and the keymask. Since the keymask and the encryption key are the same size, cipherKeyLenInBytes should be set to 2 x the AES encryption key length. For the AES-XTS mode of

operation:
以字节表示的密钥长度。对于 AES，它可以是 128 位(16 字节)、192 位(24 字节)或 256 位(32 字节)。对于 CCM 工作模式，唯一支持的密钥长度为 128 位(16 字节)。对于 CPA_CY_SYM_CIPHER_AES_F8 操作模式，cipherKeyLenInBytes 应设置为加密密钥和密钥掩码的组合长度。由于密钥掩码和加密密钥的大小相同，因此 cipherKeyLenInBytes 应设置为 AES 加密密钥长度的 2 倍。对于 AES-XTS 操作模式:

- Two keys must be provided and cipherKeyLenInBytes refers to total length of the two keys.
- 必须提供两个密钥，cipherKeyLenInBytes 是指两个密钥的总长度。
- Each key can be either 128 bits (16 bytes) or 256 bits (32 bytes).
- 每个密钥可以是 128 位(16 字节)或 256 位(32 字节)。
- Both keys must have the same size.
- 两把钥匙的大小必须相同。

CpaCySymCipherSetupData.pCipherKey
Cipher key For the CPA_CY_SYM_CIPHER_AES_F8 mode of operation, pCipherKey will point to a concatenation of the AES encryption key followed by a keymask. As per RFC3711, the keymask should be padded with trailing bytes to match the length of the encryption key used. For AES-XTS mode of operation, two keys must be provided and pCipherKey must point to the two keys concatenated together (Key1 || Key2). cipherKeyLenInBytes will contain the total size of both keys.
对于 CPA_CY_SYM_CIPHER_AES_F8 操作模式，pCipherKey 将指向 AES 加密密钥和密钥掩码的串联。根据 RFC3711，keymask 应该填充尾部字节，以匹配所使用的加密密钥的长度。对于 AES-XTS 操作模式，必须提供两个密钥，并且 pCipherKey 必须指向连接在一起的两个密钥(Key1 || Key2)。cipherKeyLenInBytes 将包含两个密钥的总大小。

CpaCySymCipherDirection CpaCySymCipherSetupData.cipherDirection
This parameter determines if the cipher operation is an encrypt or a decrypt operation. For the RC4 algorithm and the F8/CTR modes, only encrypt operations are valid.
此参数确定加密操作是加密操作还是解密操作。对于 RC4 算法和 F8/CTR 模式，只有加密操作是有效的。

---

### 8.10.3 _CpaCySymHashNestedModeSetupData Struct Reference

### 8.10.4 _ CpaCySymHashNestedModeSetupData 结构引用

8.8.2 _CpaCySymHashNestedModeSetupData Struct Reference

8.8.3_ CpaCySymHashNestedModeSetupData 结构引用

**8.8.3.1 Detailed Description**

**8.8.3.2 详细描述**

Hash Mode Nested Setup Data.
哈希模式嵌套安装数据。

This structure contains data relating to a hash session in CPA_CY_SYM_HASH_MODE_NESTED mode.
此结构包含与 CPA_CY_SYM_HASH_MODE_NESTED 模式下的哈希会话相关的数据。

**8.8.3.3 Data Fields**
**8.8.3.4 数据字段**

- **Cpa8U * pInnerPrefixData**
- Cpa8U *pInnerPrefixData
- **Cpa32U innerPrefixLenInBytes**
- Cpa32U innerPrefixLenInBytes
- **CpaCySymHashAlgorithm outerHashAlgorithm**
- CpaCySymHashAlgorithm outerHashAlgorithm
- **Cpa8U * pOuterPrefixData**
- Cpa8U *pOuterPrefixData
- **Cpa32U outerPrefixLenInBytes**
- Cpa32U outerPrefixLenInBytes

**8.8.3.5 Field Documentation**
**8.8.3.6 现场文件**

Cpa8U* CpaCySymHashNestedModeSetupData::pInnerPrefixData

A pointer to a buffer holding the Inner Prefix data. For optimal performance the prefix data SHOULD be 8-byte aligned. This data is prepended to the data being hashed before the inner hash operation is performed.
指向保存内部前缀数据的缓冲区的指针。为了获得最佳性能，前缀数据应该 8 字节对齐。在执行内部哈希操作之前，该数据将被添加到被哈希的数据之前。

Cpa32U CpaCySymHashNestedModeSetupData::innerPrefixLenInBytes

The inner prefix length in bytes. The maximum size the prefix data can be is 255 bytes.
以字节为单位的内部前缀长度。前缀数据的最大大小可以是 255 字节。

CpaCySymHashAlgorithm CpaCySymHashNestedModeSetupData::outerHashAlgorithm

The hash algorithm used for the outer hash. Note: The inner hash algorithm is provided in the hash context.
用于外部哈希的哈希算法。注意:内部哈希算法在哈希上下文中提供。

Cpa8U* CpaCySymHashNestedModeSetupData::pOuterPrefixData

A pointer to a buffer holding the Outer Prefix data. For optimal performance the prefix data SHOULD be 8-byte aligned. This data is prepended to the output from the inner hash operation before the outer hash operation is performed.
指向保存外部前缀数据的缓冲区的指针。为了获得最佳性能，前缀数据应该 8 字节对齐。在执行外部哈希运算之前，该数据将被添加到内部哈希运算的输出中。

Cpa32U CpaCySymHashNestedModeSetupData::outerPrefixLenInBytes

The outer prefix length in bytes. The maximum size the prefix data can be is 255 bytes.
以字节为单位的外部前缀长度。前缀数据的最大大小可以是 255 字节。

## 8.8.4 _CpaCySymHashAuthModeSetupData Struct Reference

## 8.8.5 _CpaCySymHashAuthModeSetupData 结构引用

### 8.8.5.1 Detailed Description
### 8.8.5.2 详细描述

Hash Auth Mode Setup Data.
哈希身份验证模式设置数据。

This structure contains data relating to a hash session in CPA_CY_SYM_HASH_MODE_AUTH mode.
此结构包含与 CPA _ CY _ SYM _哈希_MODE_AUTH 模式下的哈希会话相关的数据。

### 8.8.5.3 Data Fields
### 8.8.5.4 数据字段

- **Cpa8U * authKey**
- Cpa8U *authKey
- **Cpa32U authKeyLenInBytes**
- Cpa32U authKeyLenInBytes
- **Cpa32U aadLenInBytes**
- Cpa32U aadLenInBytes

**8.8.5.5 Field Documentation**

**8.8.5.6 现场文件**

Authentication key pointer. For the GCM (**CPA_CY_SYM_HASH_AES_GCM**) and CCM (**CPA_CY_SYM_HASH_AES_CCM**) modes of operation, this field is ignored; the authentication key is the same as the cipher key (see the field pCipherKey in struct **CpaCySymCipherSetupData**).
身份验证密钥指针。对于
GCM(CPA_CY_SYM_HASH_AES_GCMCPA_CY_SYM_HASH_AES_CCMCpaCySymCipherSetupData

Length of the authentication key in bytes. The key length MUST be less than or equal to the block size of the algorithm. It is the client's responsibility to ensure that the key length is compliant with the standard being used (for example RFC 2104, FIPS 198a).
身份验证密钥的长度，以字节为单位。密钥长度必须小于或等于算法的块大小。客户有责任确保密钥长度符合正在使用的标准(例如 RFC 2104、FIPS 198a)。

For the GCM (**CPA_CY_SYM_HASH_AES_GCM**) and CCM (**CPA_CY_SYM_HASH_AES_CCM**) modes of operation, this field is ignored; the authentication key is the same as the cipher key, and so is its length (see the field cipherKeyLenInBytes in struct **CpaCySymCipherSetupData**).
对于 GCM(CPA_CY_SYM_HASH_AES_GCMCPA_CY_SYM_HASH_AES_CCMCpaCySymCipherSetupData

The length of the additional authenticated data (AAD) in bytes. The maximum permitted value is 240 bytes, unless otherwise specified below.
附加认证数据(AAD)的长度，以字节为单位。除非下面另有说明，否则最大允许值为 240 字节。

This field must be specified when the hash algorithm is one of the following:
当哈希算法是下列算法之一时，必须指定该字段:

- For SNOW3G (**CPA_CY_SYM_HASH_SNOW3G_UIA2**), this is the length of the IV (which should be 16).
- 对于 SNOW3G(CPA_CY_SYM_HASH_SNOW3G_UIA2
- For GCM (**CPA_CY_SYM_HASH_AES_GCM**). In this case, this is the length of the Additional Authenticated Data (called A, in NIST SP800-38D).
- 对于 GCM(CPA_CY_SYM_HASH_AES_GCM
- For CCM (**CPA_CY_SYM_HASH_AES_CCM**). In this case, this is the length of the associated data (called A, in NIST SP800-38C). Note that this does NOT include the length of any padding, or the 18 bytes reserved at the start of the above field to store the block B0 and the encoded length. The maximum permitted value in this case is 222 bytes.
- 对于 CCM(CPA_CY_SYM_HASH_AES_CCM

**Note:**

**注意:**

For AES-GMAC

(**CPA_CY_SYM_HASH_AES_GMAC**) mode of operation this field is not used and should be set to 0. Instead the length of the AAD data is specified in the messageLenToHashInBytes field of the CpaCySymOpData structure.
对于 AES-GMAC(CPA_CY_SYM_HASH_AES_GMAC

---

### 8.8.5 _CpaCySymHashSetupData Struct Reference

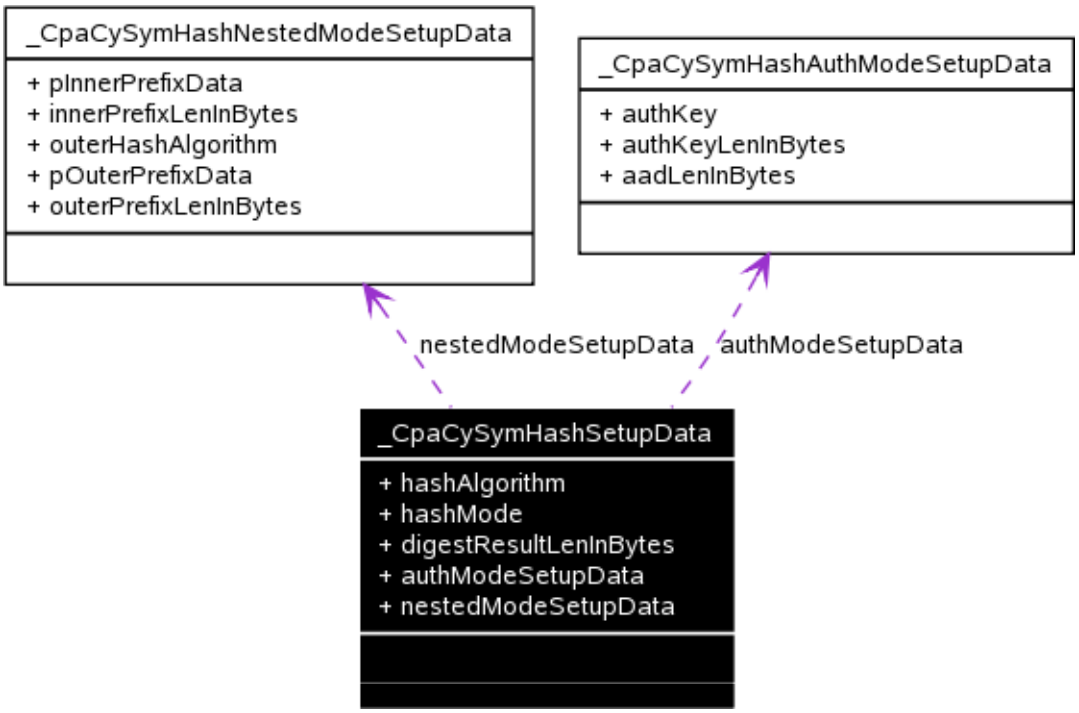
### 8.8.6 _CpaCySymHashSetupData 结构引用

Collaboration diagram for _CpaCySymHashSetupData:

_CpaCySymHashSetupData 的协作图:

8.8.4 _CpaCySymHashSetupData Struct Reference

8.8.5 _CpaCySymHashSetupData 结构引用




**8.8.5.1 Detailed Description**
**8.8.5.2 详细描述**

Hash Setup Data.
哈希设置数据。

This structure contains data relating to a hash session. The fields hashAlgorithm, hashMode and digestResultLenInBytes are common to all three hash modes and MUST be set for each mode.
该结构包含与哈希会话相关的数据。字段 hashAlgorithm、hashMode 和 digestResultLenInBytes 对于所有三种哈希模式都是通用的，并且必须针对每种模式进行设置。

**8.8.5.3 Data Fields**
**8.8.5.4 数据字段**

- **CpaCySymHashAlgorithm hashAlgorithm**
- CpaCySymHashAlgorithm hashAlgorithm
- **CpaCySymHashMode hashMode**
- CpaCySymHashMode hashMode
- **Cpa32U digestResultLenInBytes**
- Cpa32U digestResultLenInBytes
- **CpaCySymHashAuthModeSetupData authModeSetupData**
- CpaCySymHashAuthModeSetupData authModeSetupData
- **CpaCySymHashNestedModeSetupData nestedModeSetupData**
- CpaCySymHashNestedModeSetupData nestedModeSetupData

**8.8.5.5 Field Documentation**
**8.8.5.6 现场文件**

**CpaCySymHashAlgorithm _CpaCySymHashSetupData::hashAlgorithm**

CpaCySymHashAlgorithm _CpaCySymHashSetupData::hashAlgorith

Hash algorithm. For mode CPA_CY_SYM_MODE_HASH_NESTED, this is the inner hash algorithm.

哈希算法。对于 CPA_CY_SYM_MODE_HASH_NESTED 模式，这是内部哈希算法。

**CpaCySymHashMode _CpaCySymHashSetupData::hashMode**

CpaCySymHashMod_CpaCySymHashSetupData::hashMod

Mode of the hash operation. Valid options include plain, auth or nested hash mode.

哈希操作的模式。有效选项包括普通、授权或嵌套哈希模式。

**Cpa32U _CpaCySymHashSetupData::digestResultLenInBytes**

Length of the digest to be returned. If the verify option is set, this specifies the length of the digest to be compared for the session.

Cpa32_CpaCySymHashSetupData::digestResultLenInByte

For CCM (**CPA_CY_SYM_HASH_AES_CCM**), this is the octet length of the MAC, which can be one of 4, 6, 8, 10, 12, 14 or 16.

对于 CCM(CPA_CY_SYM_HASH_AES_CCM

For GCM (**CPA_CY_SYM_HASH_AES_GCM**), this is the length in bytes of the authentication tag.

对于 GCM(CPA_CY_SYM_HASH_AES_GCM

If the value is less than the maximum length allowed by the hash, the result shall be truncated. If the value is greater than the maximum length allowed by the hash, an error (**CPA_STATUS_INVALID_PARAM**) is returned from the function **cpaCySymInitSession**.
如果值小于哈希所允许的最大长度，结果将被截断。如果该值大于哈希所允许的最大长度，则会出现错误(CPA_STATUS_INVALID_PARAMcpaCySymInitSession

In the case of nested hash, it is the outer hash which determines the maximum length allowed.
在嵌套散列的情况下，是外部散列决定了允许的最大长度。

**CpaCySymHashAuthModeSetupData _CpaCySymHashSetupData::authModeSetupData**
CpaCySymHashAuthModeSetupDat_CpaCySymHashSetupData::authModeSetupDat

Authentication Mode Setup Data. Only valid for mode CPA_CY_SYM_MODE_HASH_AUTH
认证模式设置数据。仅对 CPA_CY_SYM_MODE_HASH_AUTH 模式有效

**CpaCySymHashNestedModeSetupData _CpaCySymHashSetupData::nestedModeSetupData**
CpaCySymHashNestedModeSetupDat_CpaCySymHashSetupData::nestedModeSetupDat

Nested Hash Mode Setup Data Only valid for mode CPA_CY_SYM_MODE_HASH_NESTED
嵌套散列模式设置数据仅对 CPA_CY_SYM_MODE_HASH_NESTED 模式有效

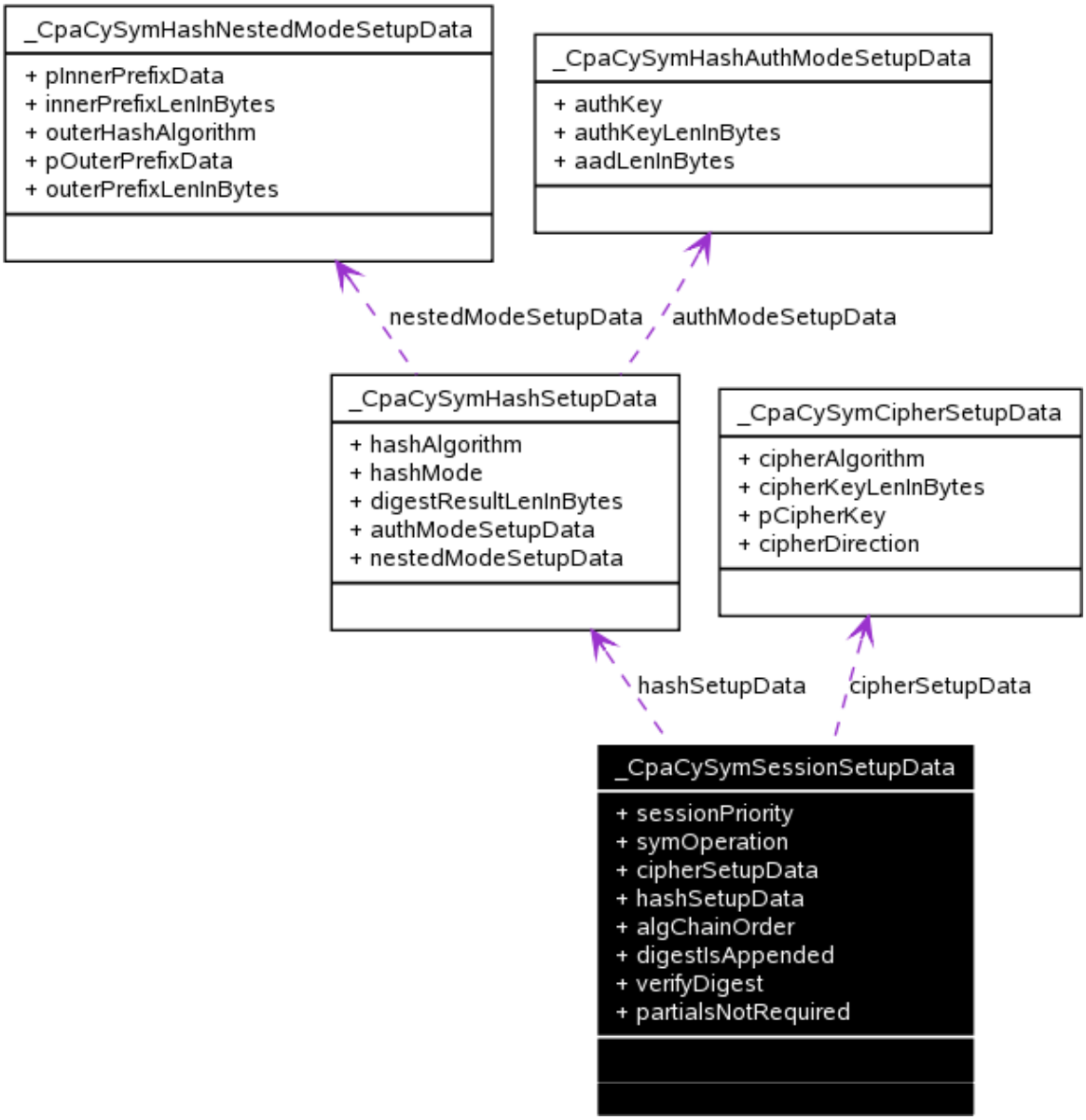
---

## 8.8.5 _CpaCySymSessionSetupData Struct Reference

## 8.8.5 _CpaCySymSessionSetupData 结构引用

Collaboration diagram for _CpaCySymSessionSetupData:

_CpaCySymSessionSetupData 的协作图:


_CpaCySymHashNestedModeSetupData
+ pInnerPrefixData
+ innerPrefixLenInBytes
+ outerHashAlgorithm
+ pOuterPrefixData
+ outerPrefixLenInBytes
_CpaCySymHashAuthModeSetupData
+ authKey
+ authKeyLenInBytes
+ aadLenInBytes
nestedModeSetupData
authModeSetupData
_CpaCySymHashSetupData
+ hashAlgorithm
+ hashMode
+ digestResultLenInBytes
+ authModeSetupData
+ nestedModeSetupData
_CpaCySymCipherSetupData
+ cipherAlgorithm
+ cipherKeyLenInBytes
+ pCipherKey
+ cipherDirection
hashSetupData
cipherSetupData
_CpaCySymSessionSetupData
+ sessionPriority
+ symOperation
+ cipherSetupData
+ hashSetupData
+ algChainOrder
+ digestIsAppended
+ verifyDigest
+ partialsNotRequired

8.8.5 _CpaCySymSessionSetupData Struct Reference

8.8.6 _CpaCySymSessionSetupData 结构引用

### 8.8.6.1 Detailed Description

### 8.8.6.2 详细描述

Session Setup Data.
会话设置数据。

This structure contains data relating to setting up a session. The client needs to complete the information in this structure in order to setup a session.
该结构包含与建立会话相关的数据。客户端需要完成该结构中的信息，以便建立会话。

### 8.8.6.3 Data Fields
### 8.8.6.4 数据字段

- **CpaCyPriority sessionPriority**
- CpaCyPriority sessionPriority
- **CpaCySymOp symOperation**
- CpaCySymOp symOperation
- **CpaCySymCipherSetupData cipherSetupData**
- CpaCySymCipherSetupData cipherSetupData
- **CpaCySymHashSetupData hashSetupData**
- CpaCySymHashSetupData hashSetupData
- **CpaCySymAlgChainOrder algChainOrder**
- CpaCySymAlgChainOrder algChainOrder
- **CpaBoolean digestIsAppended**
- CpaBoolean digestIsAppended
- **CpaBoolean verifyDigest**
- CpaBoolean verifyDigest
- **CpaBoolean partialsNotRequired**
- CpaBoolean partialsNotRequired

### 8.8.6.5 Field Documentation
### 8.8.6.6 现场文件

CpaCyPriority CpaCySymSessionSetupData::sessionPriority

Priority of this session
本次会议的优先事项

CpaCySymOp CpaCySymSessionSetupData::symOperation

Operation to perfom
要执行的操作

CpaCySymCipherSetupData CpaCySymSessionSetupData::cipherSetupData

Cipher Setup Data for the session. This member is ignored for the CPA_CY_SYM_OP_HASH operation.
会话的加密设置数据。CPA_CY_SYM_OP_HASH 运算忽略此成员。

CpaCySymHashSetupData CpaCySymSessionSetupData::hashSetupData

Hash Setup Data for a session. This member is ignored for the CPA_CY_SYM_OP_CIPHER operation.
会话的哈希设置数据。对于 CPA_CY_SYM_OP_CIPHER 操作，该成员被忽略。

CpaCySymAlgChainOrder CpaCySymSessionSetupData::algChainOrder

If this operation data structure relates to an algorithm chaining session then this parameter determines the order in which the chained operations are performed. If this structure does not relate to an algorithm chaining session then this parameter will be ignored.
如果该操作数据结构与算法链接会话相关，则该参数确定执行链接操作的顺序。如果该结构与算法链会话无关，则该参数将被忽略。

**Note:**
**注意：**

In the case of authenticated ciphers (GCM and CCM), which are also presented as "algorithm chaining", this value is also ignored. The chaining order is defined by the authenticated cipher, in those cases.
在认证密码(GCM 和 CCM)的情况下，也表示为“算法链”，该值也被忽略。在这些情况下，链接顺序由认证密码定义。

Flag indicating whether the digest is appended immediately following the region over which the digest is computed. This is true for both IPsec packets and SSL/TLS records.
指示摘要是否紧跟在计算摘要的区域后面的标志。对于 IPsec 数据包和 SSL/TLS 记录都是如此。

If this flag is set, then the value of the pDigestResult field of the structure **CpaCySymOpData** is ignored.
如果设置了此标志，则结构的 pDigestResult 字段的值 CpaCySymOpData

**Note:**
**注意：**

The value of this field is ignored for the authenticated cipher AES_CCM as the digest must be appended in this case.
对于认证密码 AES_CCM，该字段的值被忽略，因为在这种情况下必须附加摘要。

Setting digestIsAppended for hash only operations when verifyDigest is also set is not supported. For hash only operations when verifyDigest is set, digestIsAppended should be set to CPA_FALSE.

当还设置了 verifyDigest 时，不支持为仅哈希操作设置 digestIsAppended。对于设置了 verifyDigest 时的仅哈希操作，digestIsAppended 应设置为 CPA_FALSE。

CpaBoolean CpaCySymSessionSetupData::verifyDigest

This flag is relevant only for operations which generate a message digest. If set to true, the computed digest will not be written back to the buffer location specified by other parameters, but instead will be verified (i.e. compared to the value passed in at that location). The number of bytes to be written or compared is indicated by the digest output length for the session.
该标志仅与生成消息摘要的操作相关。如果设置为 true，计算出的摘要将不会被写回到由其他参数指定的缓冲区位置，而是将被验证(即，与在该位置传入的值进行比较)。要写入或比较的字节数由会话的摘要输出长度指示。

**Note:**
**注意:**

This option is only valid for full packets and for final partial packets when using partials without algorithm chaining.
当使用没有算法链接的部分包时，此选项仅对完整包和最终部分包有效。

The value of this field is ignored for the authenticated ciphers (AES_CCM and AES_GCM). Digest verification is always done for these (when the direction is decrypt) and unless the DP API is used, the message buffer will be zeroed if verification fails. When using the DP API, it is the API clients responsibility to clear the message buffer when digest verification fails.
对于认证密码(AES_CCM 和 AES_GCM ),该字段的值被忽略。对这些总是进行摘要验证(当方向是 decrypt 时),除非使用 DP API，否则如果验证失败，消息缓冲区将被清零。使用 DP API 时，当摘要验证失败时，清除消息缓冲区是 API 客户端的责任。

CpaBoolean CpaCySymSessionSetupData::partialsNotRequired

This flag indicates if partial packet processing is required for this session. If set to true, partial packet processing will not be enabled for this session and any calls to **cpaCySymPerformOp()** with the packetType parameter set to a value other than CPA_CY_SYM_PACKET_TYPE_FULL will fail.
此标志表示此会话是否需要部分数据包处理。如果设置为 true，将不会为此会话和对的任何调用启用部分数据包处理 cpaCySymPerformOp()

---

## 8.8.6 _CpaCySymSessionUpdateData Struct Reference

## 8.8.7 _CpaCySymSessionUpdateData 结构引用

### 8.8.7.1 Detailed Description
### 8.8.7.2 详细描述

Session Update Data.
会话更新数据。

This structure contains data relating to resetting a session.
该结构包含与重置会话相关的数据。

### 8.8.7.3 Data Fields
### 8.8.7.4 数据字段

- **Cpa32U flags**
- Cpa32U flags
- **Cpa8U * pCipherKey**
- Cpa8U *pCipherKey
- **CpaCySymCipherDirection cipherDirection**
- CpaCySymCipherDirection cipherDirection
- **Cpa8U * authKey**
- Cpa8U *authKey

### 8.8.7.5 Field Documentation
### 8.8.7.6 现场文件

Cpa32U CpaCySymSessionUpdateData::flags

Flags indicating which fields to update. All bits should be set to 0 except those fields to be updated.
指示要更新哪些字段的标志。除了那些要更新的字段之外，所有的位都应该设置为0。

Cpa8U* CpaCySymSessionUpdateData::pCipherKey

Cipher key. The same restrictions apply as described in the corresponding field of the data structure **CpaCySymCipherSetupData**.
密码钥匙。如在数据结构的相应字段中所描述的，同样的限制也适用 CpaCySymCipherSetupData。

CpaCySymCipherDirection CpaCySymSessionUpdateData::cipherDirection

This parameter determines if the cipher operation is an encrypt or a decrypt operation. The same restrictions apply as described in the corresponding field of the data structure **CpaCySymCipherSetupData**.
此参数确定加密操作是加密操作还是解密操作。如在数据结构的相应字段中所描述的，同样的限制也适用 CpaCySymCipherSetupData

**Cpa8U* CpaCySymSessionUpdateData::authKey**

Authentication key pointer. The same restrictions apply as described in the corresponding field of the data structure **CpaCySymHashAuthModeSetupData**.
身份验证密钥指针。如在数据结构的相应字段中所描述的，同样的限制也适用 CpaCySymHashAuthModeSetupData

---

## 8.8.8 _CpaCySymOpData Struct Reference

## 8.8.9 _CpaCySymOpData 结构引用

### 8.8.9.1 Detailed Description
### 8.8.9.2 详细描述

Cryptographic Component Operation Data.
加密组件操作数据。

This structure contains data relating to performing cryptographic processing on a data buffer. This request is used with **cpaCySymPerformOp()** call for performing cipher, hash, auth cipher or a combined hash and cipher operation.
该结构包含与在数据缓冲区上执行加密处理相关的数据。此请求与一起使用 cpaCySymPerformOp()

**See also:**
**另请参见:**

**CpaCySymPacketType**
CpaCySymPacketType

**Note:**

**注意:**

8.8.7 CpaCySymOpData Struct

If the client modifies or frees the memory referenced in this structure after it has been submitted to the cpaCySymPerformOp function, and before it has been returned in the callback, undefined behavior will result.
如果客户端在将此结构中引用的内存提交给 cpaCySymPerformOp 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。

**8.8.9.3 Data Fields**

**8.8.9.4 数据字段**

- **CpaCySymSessionCtx sessionCtx**
- CpaCySymSessionCtx sessionCtx
- **CpaCySymPacketType packetType**
- CpaCySymPacketType packetType
- **Cpa8U * pIv**
- Cpa8U *pIv
- **Cpa32U ivLenInBytes**
- Cpa32U ivLenInBytes
- **Cpa32U cryptoStartSrcOffsetInBytes**
- Cpa32U cryptoStartSrcOffsetInBytes
- **Cpa32U messageLenToCipherInBytes**
- Cpa32U messageLenToCipherInBytes
- **Cpa32U hashStartSrcOffsetInBytes**
- Cpa32U hashStartSrcOffsetInBytes
- **Cpa32U messageLenToHashInBytes**
- Cpa32U messageLenToHashInBytes
- **Cpa8U * pDigestResult**
- Cpa8U *pDigestResult
- **Cpa8U * pAdditionalAuthData**
- Cpa8U *pAdditionalAuthData

**8.8.9.5 Field Documentation**

**8.8.9.6 现场文件**

CpaCySymSessionCtx CpaCySymOpData::sessionCtx

Handle for the initialized session context
初始化的会话上下文的句柄

CpaCySymPacketType CpaCySymOpData::packetType

Selects the packet type
选择数据包类型

Cpa8U* CpaCySymOpData::pIv

Initialization Vector or Counter.
初始化向量或计数器。

- For block ciphers in CBC or F8 mode, or for Kasumi in F8 mode, or for SNOW3G in UEA2 mode, this is the Initialization Vector (IV) value.
- 对于 CBC 或 F8 模式下的块密码，或 F8 模式下的小霞，或 UEA2 模式下的 SNOW3G，这是初始化向量 (IV) 值。
- For block ciphers in CTR mode, this is the counter.
- 对于 CTR 模式下的分组密码，这是计数器。
- For GCM mode, this is either the IV (if the length is 96 bits) or J0 (for other sizes), where J0 is as

defined by NIST SP800-38D. Regardless of the IV length, a full 16 bytes needs to be allocated.

- 对于 GCM 模式，这是 IV(如果长度为 96 位)或 J0(对于其他尺寸)，其中 J0 由 NIST SP800-38D 定义。无论 IV 长度如何，都需要分配整整 16 个字节。
- For CCM mode, the first byte is reserved, and the nonce should be written starting at &pIv[1] (to
- 对于 CCM 模式，第一个字节是保留的，nonce 应该从&pIv[1]开始写入(到

allow space for the implementation to write in the flags in the first byte). Note that a full 16 bytes should be allocated, even though the ivLenInBytes field will have a value less than this. The macro **CPA_CY_SYM_CCM_SET_NONCE** may be used here.

允许实现在第一个字节中写入标志的空间)。请注意，应该分配完整的 16 个字节，即使 ivLenInBytes 字段的值小于该值。宏 CPA_CY_SYM_CCM_SET_NONCE

- For AES-XTS, this is the 128bit tweak, i, from IEEE Std 1619-2007.
- 对于 AES-XTS，这是来自 IEEE Std 1619-2007 的 128 位 tweak I。

For optimum performance, the data pointed to SHOULD be 8-byte aligned.

The IV/Counter will be updated after every partial cryptographic operation.

为了获得最佳性能，指向的数据应该 8 字节对齐。IV/计数器将在每次部分

加密操作后更新。

### Cpa32U _CpaCySymOpData::ivLenInBytes

Cpa32_CpaCySymOpData::ivLenInByte

Length of valid IV data pointed to by the pIv parameter.

pIv 参数指向的有效 IV 数据的长度。

- For block ciphers in CBC or F8 mode, or for Kasumi in F8 mode, or for SNOW3G in UEA2 mode, this is the length of the IV (which must be the same as the block length of the cipher).
- 对于 CBC 或 F8 模式的分组密码，或 F8 模式的小霞，或 UEA2 模式的 SNOW3G，这是 IV 的长度(必须与密码的分组长度相同)。
- For block ciphers in CTR mode, this is the length of the counter (which must be the same as the block length of the cipher).
- 对于 CTR 模式下的分组密码，这是计数器的长度(必须与密码的分组长度相同)。
- For GCM mode, this is either 12 (for 96-bit IVs) or 16, in which case pIv points to J0.
- 对于 GCM 模式，这是 12(对于 96 位 IVs)或 16，在这种情况下 pIv 指向 J0。
- For CCM mode, this is the length of the nonce, which can be in the range 7 to 13 inclusive.
- 对于 CCM 模式，这是随机数的长度，可以在 7 到 13 的范围内，包括 7 和 13。

### Cpa32U _CpaCySymOpData::cryptoStartSrcOffsetInBytes

Cpa32_CpaCySymOpData::cryptoStartSrcOffsetInByte

Starting point for cipher processing, specified as number of bytes from start of data in the source buffer. The result of the cipher operation will be written back into the output buffer starting at this location.

密码处理的起点，指定为从源缓冲区中数据开始的字节数。加密操作的结果将从该位置开始写回输出缓冲区。

### Cpa32U _CpaCySymOpData::messageLenToCipherInBytes

The message length, in bytes, of the source buffer on which the cryptographic operation will be computed. This must be a multiple of the block size if a block cipher is being used. This is also the same as the result

Cpa32_CpaCySymOpData::messageLenToCipherInByte

length.

长度。

**Note:**

**注意:**

In the case of CCM (**CPA_CY_SYM_HASH_AES_CCM**), this value should not include the length of

the padding or the length of the MAC; the driver will compute the actual number of bytes over which the encryption will occur, which will include these values.
在 CCM 的情况下(CPA_CY_SYM_HASH_AES_CCM

There are limitations on this length for partial operations. Refer to the cpaCySymPerformOp function description for details.
对于部分操作，这个长度是有限制的。有关详细信息，请参考 cpaCySymPerformOp 函数描述。

On some implementations, this length may be limited to a 16-bit value (65535 bytes).

For AES-GMAC (**CPA_CY_SYM_HASH_AES_GMAC**), this field should be set to 0.

在一些实施方式中，该长度可能被限制为 16 位值(65535 字节)。对于 AES-

GMAC(CPA_CY_SYM_HASH_AES_GMAC

### Cpa32U _CpaCySymOpData::hashStartSrcOffsetInBytes

Cpa32_CpaCySymOpData::hashStartSrcOffsetInByte

Starting point for hash processing, specified as number of bytes from start of packet in source buffer.
哈希处理的起点，指定为从源缓冲区中的数据包开始算起的字节数。

**Note:**
**注意:**

For CCM and GCM modes of operation, this field is ignored. The field **pAdditionalAuthData** field should be set instead.
对于 CCM 和 GCM 工作模式，该域被忽略。田野 pAdditionalAuthData

For AES-GMAC (**CPA_CY_SYM_HASH_AES_GMAC**) mode of operation, this field specifies the start of the AAD data in the source buffer.
对于 AES-GMAC(CPA_CY_SYM_HASH_AES_GMAC

### Cpa32U _CpaCySymOpData::messageLenToHashInBytes

Cpa32_CpaCySymOpData::messageLenToHashInByte

The message length, in bytes, of the source buffer that the hash will be computed on.
将计算哈希的源缓冲区的消息长度(以字节为单位)。

**Note:**
**注意:**

There are limitations on this length for partial operations. Refer to the **cpaCySymPerformOp**

对于部分操作，这个长度是有限制的。请参考 cpaCySymPerformOp
function description for details.
详细功能描述。

For CCM and GCM modes of operation, this field is ignored. The field **pAdditionalAuthData** field should be set instead.
对于 CCM 和 GCM 工作模式，该域被忽略。田野 pAdditionalAuthData

For AES-GMAC (**CPA_CY_SYM_HASH_AES_GMAC**) mode of operation, this field specifies the length of the AAD data in the source buffer. The maximum length supported for AAD data for AES-GMAC is 16383 bytes.
对于 AES-GMAC(CPA_CY_SYM_HASH_AES_GMAC

On some implementations, this length may be limited to a 16-bit value (65535 bytes).
在一些实施方式中，该长度可能被限制为 16 位值(65535 字节)。

**Cpa8U* _CpaCySymOpData::pDigestResult**

If the digestIsAppended member of the **CpaCySymSessionSetupData** structure is NOT set then this is a pointer to the location where the digest result should be inserted (in the case of digest generation) or where
Cpa8U_CpaCySymOpData::pDigestResulCpaCySymSessionSetupData
the purported digest exists (in the case of digest verification).
声称的摘要存在(在摘要验证的情况下)。

At session registration time, the client specified the digest result length with the digestResultLenInBytes member of the **CpaCySymHashSetupData** structure. The client must allocate at least digestResultLenInBytes of physically contiguous memory at this location.
在会话注册时，客户端用 CpaCySymHashSetupData

For partial packet processing without algorithm chaining, this pointer will be ignored for all but the final partial operation.
对于没有算法链接的部分包处理，除了最后的部分操作，该指针将被忽略。

For digest generation, the digest result will overwrite any data at this location.
对于摘要生成，摘要结果将覆盖此位置的任何数据。

**Note:**
**注意:**

For GCM (**CPA_CY_SYM_HASH_AES_GCM**), for "digest result" read "authentication tag T".
对于 GCM(CPA_CY_SYM_HASH_AES_GCM

If the digestIsAppended member of the **CpaCySymSessionSetupData** structure is set then this value is ignored and the digest result is understood to be in the destination buffer for digest generation, and in the source buffer for digest verification. The location of the digest result in this case is immediately following the region over which the digest is computed.
如果 CpaCySymSessionSetupData

**Cpa8U* _CpaCySymOpData::pAdditionalAuthData**

Pointer to Additional Authenticated Data (AAD) needed for authenticated cipher mechanisms (CCM and GCM), and to the IV for SNOW3G authentication (**CPA_CY_SYM_HASH_SNOW3G_UIA2**). For other
Cpa8U_CpaCySymOpData::pAdditionalAuthDatCPA_CY_SYM_HASH_SNOW3G_UIA2
authentication mechanisms this pointer is ignored.
验证机制该指针被忽略。

The length of the data pointed to by this field is set up for the session in the **CpaCySymHashAuthModeSetupData** structure as part of the **cpaCySymInitSession** function call. This length must not exceed 240 bytes.
此字段指向的数据长度是在中为进程设置的 CpaCySymHashAuthModeSetupData cpaCySymInitSession

Specifically for CCM (**CPA_CY_SYM_HASH_AES_CCM**), the caller should setup this field as follows:
专门针对 CCM(CPA_CY_SYM_HASH_AES_CCM

- the nonce should be written starting at an offset of one byte into the array, leaving room for the implementation to write in the flags to the first byte. For example,
- nonce 应该从一个字节的偏移量开始写入数组，为实现将标志写入第一个字节留出空间。举个例子，
  memcpy(&pOpData->pAdditionalAuthData[1], pNonce, nonceLen);
  The macro **CPA_CY_SYM_CCM_SET_NONCE** may be used here.
  memcpy(& pOpData->
  pAdditionalAuthData[1], pNonce, nonceLen)；宏
  CPA_CY_SYM_CCM_SET_NONCE

- the additional authentication data itself should be written starting at an offset of 18 bytes into the array, leaving room for the length encoding in the first two bytes of the second block. For example, memcpy(&pOpData->pAdditionalAuthData[18], pAad, aadLen);
- 附加认证数据本身应该从 18 字节的偏移量开始写入数组，为第二个块的前两个字节中的长度编码留出空间。比如 memcpy(& pOpData-> pAdditionalAuthData[18], pAad, aadLen)；
  The macro **CPA_CY_SYM_CCM_SET_AAD** may be used here.
  宏 CPA_CY_SYM_CCM_SET_AAD

- the array should be big enough to hold the above fields, plus any padding to round this up to the nearest multiple of the block size (16 bytes). Padding will be added by the implementation.
- 该数组应该足够大，能够容纳上述字段，加上任何填充符，以便将其向上舍入到块大小的最近倍数(16 字节)。填充将由实现添加。

Finally, for GCM (**CPA_CY_SYM_HASH_AES_GCM**), the caller should setup this field as follows:
最后，对于 GCM(CPA_CY_SYM_HASH_AES_GCM

- the AAD is written in starting at byte 0
- AAD 从字节 0 开始写入
- the array must be big enough to hold the AAD, plus any padding to round this up to the nearest multiple of the block size (16 bytes). Padding will be added by the implementation.
- 数组必须足够大，以容纳 AAD，加上任何填充符，以将其向上舍入到块大小的最近倍数(16 字节)。填充将由实现添加。

**Note:**

**注意：**

For AES-GMAC

(**CPA_CY_SYM_HASH_AES_GMAC**) mode of operation, this field is not used and should be set to 0. Instead the AAD data should be placed in the source buffer.
对于 AES-GMAC(CPA_CY_SYM_HASH_AES_GMAC

---

## 8.8.8 _CpaCySymStats Struct Reference

## 8.8.9 _CpaCySymStats 结构引用

### 8.8.9.1 Detailed Description

Cryptographic Component Statistics.

**Deprecated:**

### 8.8.9.2 加密组件统计的详细描

述。Deprecated:

As of v1.3 of the cryptographic API, this structure has been deprecated, replaced by
从加密 API 的 1.3 版开始，这种结构已被取代，由
**CpaCySymStats64**.
CpaCySymStats64。

This structure contains statistics on the Symmetric Cryptographic operations. Statistics are set to zero when the component is initialized.
此结构包含对称加密操作的统计信息。当组件初始化时，统计信息被设置为零。

### 8.8.9.3 Data Fields
### 8.8.9.4 数据字段

- **Cpa32U numSessionsInitialized**
- Cpa32U numSessionsInitialized
- **Cpa32U numSessionsRemoved**
- Cpa32U numSessionsRemoved
- **Cpa32U numSessionErrors**
- Cpa32U numSessionErrors
- **Cpa32U numSymOpRequests**
- Cpa32U numSymOpRequests
- **Cpa32U numSymOpRequestErrors**
- Cpa32U numSymOpRequestErrors
- **Cpa32U numSymOpCompleted**
- Cpa32U numSymOpCompleted
- **Cpa32U numSymOpCompletedErrors**
- Cpa32U numSymOpCompletedErrors
- **Cpa32U numSymOpVerifyFailures**
- Cpa32U numSymOpVerifyFailures

### 8.8.9.5 Field Documentation
### 8.8.9.6 现场文件

**Cpa32U CpaCySymStats::numSessionsInitialized**

Number of session initialized
初始化的会话数

Number of sessions removed
删除的会话数

Number of session initialized and removed errors.
初始化的会话数和删除的错误数。

Number of successful symmetric operation requests.
成功的对称操作请求数。

8.8.10 _CpaCySymStats64 Struct Reference

8.8.11 _CpaCySymStats64 结构引用

Number of operation requests that had an error and could not be processed.

有错误且无法处理的操作请求数。

Cpa32U CpaCySymStats::numSymOpCompleted

Number of operations that completed successfully.

成功完成的操作数。

Cpa32U CpaCySymStats::numSymOpCompletedErrors

Number of operations that could not be completed successfully due to errors.

由于错误而无法成功完成的操作数。

Cpa32U CpaCySymStats::numSymOpVerifyFailures

Number of operations that completed successfully, but the result of the digest verification test was that it failed. Note that this does not indicate an error condition.

成功完成但摘要验证测试结果为失败的操作数。请注意，这并不表示存在错误情况。

---

## 8.8.9 _CpaCySymStats64 Struct Reference

## 8.8.10 _CpaCySymStats64 结构引用

### 8.8.10.1 Detailed Description

### 8.8.10.2 详细描述

Cryptographic Component Statistics (64-bit version).

加密组件统计信息(64 位版本)。

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

此结构包含对称加密操作的 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。

### 8.8.10.3 Data Fields

### 8.8.10.4 数据字段

- **Cpa64U numSessionsInitialized**
- Cpa64U numSessionsInitialized
- **Cpa64U numSessionsRemoved**
- Cpa64U numSessionsRemoved
- **Cpa64U numSessionErrors**
- Cpa64U numSessionErrors
- **Cpa64U numSymOpRequests**
- Cpa64U numSymOpRequests
- **Cpa64U numSymOpRequestErrors**
- Cpa64U numSymOpRequestErrors
- **Cpa64U numSymOpCompleted**
- Cpa64U numSymOpCompleted
- **Cpa64U numSymOpCompletedErrors**
- Cpa64U numSymOpCompletedErrors
- **Cpa64U numSymOpVerifyFailures**

- Cpa64U numSymOpVerifyFailures

#### 8.8.10.5 Field Documentation
#### 8.8.10.6 现场文件

Cpa64U CpaCySymStats64::numSessionsInitialized

Number of session initialized
初始化的会话数

Cpa64U CpaCySymStats64::numSessionsRemoved

Number of sessions removed
删除的会话数

Cpa64U CpaCySymStats64::numSessionErrors

Number of session initialized and removed errors.
初始化的会话数和删除的错误数。

Cpa64U CpaCySymStats64::numSymOpRequests

Number of successful symmetric operation requests.
成功的对称操作请求数。

Cpa64U CpaCySymStats64::numSymOpRequestErrors

Number of operation requests that had an error and could not be processed.
有错误且无法处理的操作请求数。

Cpa64U CpaCySymStats64::numSymOpCompleted

Number of operations that completed successfully.
成功完成的操作数。

**Cpa64U CpaCySymStats64::numSymOpCompletedErrors**

Number of operations that could not be completed successfully due to errors.
由于错误而无法成功完成的操作数。

**Cpa64U CpaCySymStats64::numSymOpVerifyFailures**

Number of operations that completed successfully, but the result of the digest verification test was that it failed. Note that this does not indicate an error condition.
成功完成但摘要验证测试结果为失败的操作数。请注意，这并不表示存在错误情况。

---

## 8.8.11 _CpaCySymCapabilitiesInfo Struct Reference

## 8.8.12 _CpaCySymCapabilitiesInfo 结构引用

### 8.8.12.1 Detailed Description
### 8.8.12.2 详细描述

Symmetric Capabilities Info
对称功能信息

This structure contains the capabilities that vary across implementations of the symmetric sub-API of the cryptographic API. This structure is used in conjunction with **cpaCySymQueryCapabilities()** to determine the capabilities supported by a particular API implementation.
该结构包含在加密 API 的对称子 API 的实现之间变化的能力。此结构与一起使用 cpaCySymQueryCapabilities()

For example, to see if an implementation supports cipher **CPA_CY_SYM_CIPHER_AES_CBC**, use the code
例如，查看实现是否支持加密 CPA_CY_SYM_CIPHER_AES_CBC

```
if  (CPA_BITMAP_BIT_TEST(capInfo.ciphers,  CPA_CY_SYM_CIPHER_AES_CBC))
如果(CPA_BITMAP_BIT_TESTCPA_CY_SYM_CIPHER_AES_CBC
{
{

    // algo is supported

}
}
else
其他
{
{


}
}
```

8.8.10 CpaCySymCapabilitiesInfo Struct

//支持算法

```
// algo is not supported
//不支持算法
```

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

客户端必须为此结构和任何需要内存的成员分配内存。当结构被传递给函数时，内存的所有权就传递给了函数。当函数返回时，内存的所有权返回给客户端。

### 8.8.12.3 Public Member Functions
### 8.8.12.4 公共成员函数

- **CPA_BITMAP** (ciphers, CPA_CY_SYM_CIPHER_CAP_BITMAP_SIZE)
- CPA_BITMAP（密码，CPA _ CY _ SYM _密码_ CAP _位图_大小）
- **CPA_BITMAP** (hashes, CPA_CY_SYM_HASH_CAP_BITMAP_SIZE)
- CPA_BITMAP（哈希，CPA _ CY _ SYM _哈希_ CAP _位图_大小）

### 8.8.12.5 Data Fields
### 8.8.12.6 数据字段

- **CpaBoolean partialPacketSupported**
- CpaBoolean partialPacketSupported

### 8.8.12.7 Member Function Documentation
### 8.8.12.8 成员函数文档

CpaCySymCapabilitiesInfo::CPA_BITMAP( ciphers

CpaCySymCapabilitiesInfo::CPA_BITMAP( ciphers ,

CPA_CY_SYM_CIPHER_CAP_BITMAP_SIZE

CPA_CY_SYM_CIPHER_CAP_BITMAP_SIZE

Bitmap representing which cipher algorithms (and modes) are supported by the instance. Bits can be tested using the macro **CPA_BITMAP_BIT_TEST**. The bit positions are those specified in the enumerated type **CpaCySymCipherAlgorithm**.

表示实例支持哪些密码算法(和模式)的位图。可以使用宏来测试位

CPA_BITMAP_BIT_TESTCpaCySymCipherAlgorithm

_CpaCySymCapabilitiesInfo::CPA_BITMAP( hashes

_CpaCySymCapabilitiesInfo::CPA_BITMAP( hashes ,

CPA_CY_SYM_HASH_CAP_BITMAP_SIZE

CPA_CY_SYM_HASH_CAP_BITMAP_SIZE

8.8.10 CpaCySymCapabilitiesInfo Struct

Bitmap representing which hash/authentication algorithms are supported by the instance. Bits can be tested using the macro **CPA_BITMAP_BIT_TEST**. The bit positions are those specified in the enumerated type **CpaCySymHashAlgorithm**.

表示实例支持哪些哈希/身份验证算法的位图。可以使用宏来测试位
CPA_BITMAP_BIT_TESTCpaCySymHashAlgorithm

**8.8.12.9 Field Documentation**

**8.8.12.10 现场文件**

CpaBoolean CpaCySymCapabilitiesInfo::partialPacketSupported

CPA_TRUE if instance supports partial packets. See **CpaCySymPacketType**.
如果实例支持部分数据包，则 CPA_TRUE。看见 CpaCySymPacketType

---

## 8.9 Define Documentation

## 8.10 定义文档

#define CPA_CY_SYM_CIPHER_CAP_BITMAP_SIZE

Size of bitmap needed for cipher "capabilities" type.
密码“功能”类型所需的位图大小。

Defines the number of bits in the bitmap to represent supported ciphers in the type **CpaCySymCapabilitiesInfo**. Should be set to at least one greater than the largest value in the enumerated type **CpaCySymHashAlgorithm**, so that the value of the enum constant can also be used as the bit position in the bitmap.
定义位图中表示该类型中支持的密码的位数 CpaCySymCapabilitiesInfoCpaCySymHashAlgorithm

A larger value was chosen to allow for extensibility without the need to change the size of the bitmap (to ease backwards compatibility in future versions of the API).
选择较大的值是为了在不需要更改位图大小的情况下实现可扩展性(以便在 API 的未来版本中易于向后兼容)。

#define CPA_CY_SYM_HASH_CAP_BITMAP_SIZE

Size of bitmap needed for hash "capabilities" type.
哈希“功能”类型所需的位图大小。

Defines the number of bits in the bitmap to represent supported hashes in the type **CpaCySymCapabilitiesInfo**. Should be set to at least one greater than the largest value in the enumerated type **CpaCySymHashAlgorithm**, so that the value of the enum constant can also be used as the bit position in the bitmap.
定义位图中表示该类型中支持的哈希的位数 CpaCySymCapabilitiesInfoCpaCySymHashAlgorithm

A larger value was chosen to allow for extensibility without the need to change the size of the bitmap (to ease backwards compatibility in future versions of the API).
选择较大的值是为了在不需要更改位图大小的情况下实现可扩展性(以便在 API 的未来版本中易于向后兼容)。

#define CPA_CY_SYM_CCM_SET_NONCE ( pOpData, pNonce, nonceLen )

Setup the nonce for CCM.

#define CPA_CY_SYM_CCM_SET_NONCE ( pOpData, pNonce, nonceLen )

为 CCM 设置随机数。

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

该宏在的适当位置设置随机数 CpaCySymOpData CPA_CY_SYM_HASH_AES_CCM

Setup the additional authentication data for CCM.

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

为 CCM 设置附加认证数据。

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

该宏在的适当位置设置附加身份验证数据 CpaCySymOpDataCPA_CY_SYM_HASH_AES_CCM

# 8.11 Typedef Documentation

# 8.12 Typedef 文档

typedef void* **CpaCySymSessionCtx**

typedef void*CpaCySymSessionCt

Cryptographic component symmetric session context handle.

加密组件对称会话上下文句柄。

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

加密会话上下文的句柄。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 cpaCySymSessionCtxGetSizecpaCySymSessionCtxGetDynamicSize cpaCySymInitSession cpaCySymRemoveSession

typedef enum _**CpaCySymPacketType CpaCySymPacketType**

typedef 枚举_CpaCySymPacketTypCpaCySymPacketTyp

Packet type for the cpaCySymPerformOp function

cpaCySymPerformOp 函数的数据包类型

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

枚举，用于向对称加密执行函数指示需要对哪种类型的数据包调用操作。当需要对客户端可获得的多个部分的消息执行处理时(例如，由于数据包的网络碎片)，多部分密码和散列操作非常有用。

**Note:**

**注意:**

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

对于支持部分数据包处理的操作有一些限制。有关详细信息，请参见函数 cpaCySymPerformOp

**See also:**

**另请参见:**

**cpaCySymPerformOp()**

cpaCySymPerformOp()

typedef enum _**CpaCySymOp CpaCySymOp**

typedef 枚举_CpaCySymOCpaCySymO

Types of operations supported by the cpaCySymPerformOp function.

cpaCySymPerformOp 函数支持的操作类型。

This enumeration lists different types of operations supported by the cpaCySymPerformOp function. The operation type is defined during session registration and cannot be changed for a session once it has been setup.
此枚举列出了由 cpaCySymPerformOp 函数支持的不同类型的操作。操作类型是在会话注册期间定义的，一旦设置，就不能为会话更改。

**See also:**
**另请参见：**
**cpaCySymPerformOp**
cpaCySymPerformOp

typedef enum _**CpaCySymCipherAlgorithm CpaCySymCipherAlgorithm**
typedef 枚举_CpaCySymCipherAlgorithm CpaCySymCipherAlgorith

Cipher algorithms.
密码算法。

This enumeration lists supported cipher algorithms and modes.
此枚举列出了支持的密码算法和模式。

typedef enum _**CpaCySymCipherDirection CpaCySymCipherDirection**
typedef 枚举_CpaCySymCipherDirectioCpaCySymCipherDirectio

Symmetric Cipher Direction
对称密码方向

This enum indicates the cipher direction (encryption or decryption).
此枚举指示密码方向(加密或解密)。

typedef struct _**CpaCySymCipherSetupData CpaCySymCipherSetupData**
typedef 结构_CpaCySymCipherSetupDatCpaCySymCipherSetupDat

Symmetric Cipher Setup Data.
对称密码设置数据。

This structure contains data relating to Cipher (Encryption and Decryption) to set up a session.
该结构包含与建立会话的密码(加密和解密)相关的数据。

typedef enum _**CpaCySymHashMode CpaCySymHashMode**
typedef 枚举_CpaCySymHashModCpaCySymHashMod

8.10 Typedef

Symmetric Hash mode

对称散列模式

This enum indicates the Hash Mode.
此枚举指示哈希模式。

typedef enum CpaCySymHashAlgorithm CpaCySymHashAlgorithm

Hash algorithms.
哈希算法。

This enumeration lists supported hash algorithms.
此枚举列出了支持的哈希算法。

typedef struct CpaCySymHashNestedModeSetupData CpaCySymHashNestedModeSetupData

Hash Mode Nested Setup Data.
哈希模式嵌套安装数据。

This structure contains data relating to a hash session in CPA_CY_SYM_HASH_MODE_NESTED mode.
此结构包含与 CPA _ CY _ SYM _哈希_ MODE _嵌套模式中的哈希会话相关的数据。

typedef struct CpaCySymHashAuthModeSetupData CpaCySymHashAuthModeSetupData

Hash Auth Mode Setup Data.
哈希身份验证模式设置数据。

This structure contains data relating to a hash session in CPA_CY_SYM_HASH_MODE_AUTH mode.
此结构包含与 CPA _ CY _ SYM _哈希_MODE_AUTH 模式下的哈希会话相关的数据。

typedef struct CpaCySymHashSetupData CpaCySymHashSetupData

Hash Setup Data.
哈希设置数据。

This structure contains data relating to a hash session. The fields hashAlgorithm, hashMode and digestResultLenInBytes are common to all three hash modes and MUST be set for each mode.
该结构包含与哈希会话相关的数据。字段 hashAlgorithm、hashMode 和 digestResultLenInBytes 对于所有三种哈希模式都是通用的，并且必须针对每种模式进行设置。

typedef enum CpaCySymAlgChainOrder CpaCySymAlgChainOrder

Algorithm Chaining Operation Ordering
算法链接操作排序

This enum defines the ordering of operations for algorithm chaining.
此枚举定义算法链的操作顺序。

typedef struct CpaCySymSessionSetupData CpaCySymSessionSetupData

Session Setup Data.
会话设置数据。

This structure contains data relating to setting up a session. The client needs to complete the information in this structure in order to setup a session.
该结构包含与建立会话相关的数据。客户端需要完成该结构中的信息，以便建立会话。

typedef struct CpaCySymSessionUpdateData CpaCySymSessionUpdateData

Session Update Data.
会话更新数据。

This structure contains data relating to resetting a session.
该结构包含与重置会话相关的数据。

Cryptographic Component Operation Data.
加密组件操作数据。

This structure contains data relating to performing cryptographic processing on a data buffer. This request is used with **cpaCySymPerformOp()** call for performing cipher, hash, auth cipher or a combined hash and cipher operation.
该结构包含与在数据缓冲区上执行加密处理相关的数据。此请求与一起使用 cpaCySymPerformOp()

**See also:**
**另请参见:**

**CpaCySymPacketType**
CpaCySymPacketType

**Note:**

**注意:**

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

如果客户端在将此结构中引用的内存提交给 cpaCySymPerformOp 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。

## typedef struct _**CpaCySymStats CPA_DEPRECATED**

typedef 结构_CpaCySymStatCPA_DEPRECATE

Cryptographic Component Statistics.

加密组件统计。

**Deprecated:**

Deprecated:

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

从加密 API 的 1.3 版开始，这种结构已被取代，由 CpaCySymStats64。

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

此结构包含对称加密操作的统计信息。当组件初始化时，统计信息被设置为零。

## typedef struct _**CpaCySymStats64 CpaCySymStats64**

typedef 结构_CpaCySymStats6CpaCySymStats6

Cryptographic Component Statistics (64-bit version).

加密组件统计信息(64 位版本)。

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

此结构包含对称加密操作的 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。

Definition of callback function

typedef void(* **CpaCySymCbFunc**)(void *pCallbackTag, **CpaStatus** status, const **CpaCySymOp**

回调函数的定义

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

这是回调函数原型。回调函数由应用程序使用 cpaCySymInitSession() 函数调用。

**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* Opaque value provided by user while making individual function call.

[in] pCallbackTag 用户在进行单个函数调用时提供的不透明值。

[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] *operationType* Identifies the operation type that was requested in the cpaCySymPerformOp function.

[in] operationType 标识在 cpaCySymPerformOp 函数中请求的操作类型。

[in] *pOpData* Pointer to structure with input parameters.

[in]指向具有输入参数的结构的 pOpData 指针。

[in] *pDstBuffer* Caller MUST allocate a sufficiently sized destination buffer to hold the data output. For out-of-place processing the data outside the cryptographic regions in the source buffer are copied into the destination buffer. To perform "in-place" processing set the pDstBuffer parameter in cpaCySymPerformOp function to point at the same location as pSrcBuffer. For optimum performance, the data pointed to SHOULD be 8-byte aligned.

[in] pDstBuffer 调用方必须分配足够大的目标缓冲区来保存数据输出。对于错位处理，将源缓冲区中加密区域之外的数据复制到目标缓冲区中。若要执行“就地”处理，请将 cpacysymperformatopfunction 中的 pDstBuffer 参数设置为指向与 pSrcBuffer 相同的位置。为了获得最佳性能，指向的数据应该 8 字节对齐。

[in] *verifyResult* This parameter is valid when the verifyDigest option is set in the CpaCySymSessionSetupData structure. A value of CPA_TRUE indicates that the compare succeeded. A value of CPA_FALSE indicates that the compare failed for an unspecified reason.

[in] verifyResult 当在 CpaCySymSessionSetupData 结构中设置了 verifyDigest 选项时，此参数有效。CPA_TRUE 值表示比较成功。CPA_FALSE 值表示比较因未指明的原因而失败。

**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:**

**cpaCySymInitSession()**, **cpaCySymRemoveSession()**
cpaCySymInitSession(), cpaCySymRemoveSession()

typedef struct _**CpaCySymCapabilitiesInfo CpaCySymCapabilitiesInfo**
typedef 结构_CpaCySymCapabilitiesInfCpaCySymCapabilitiesInf

Symmetric Capabilities Info
对称功能信息

This structure contains the capabilities that vary across implementations of the symmetric sub-API of the cryptographic API. This structure is used in conjunction with **cpaCySymQueryCapabilities()** to determine the capabilities supported by a particular API implementation.
该结构包含在加密 API 的对称子 API 的实现之间变化的能力。此结构与一起使用 cpaCySymQueryCapabilities()

For example, to see if an implementation supports cipher **CPA_CY_SYM_CIPHER_AES_CBC**, use the code
例如，查看实现是否支持加密 CPA_CY_SYM_CIPHER_AES_CBC

```
if  (CPA_BITMAP_BIT_TEST(capInfo.ciphers,  CPA_CY_SYM_CIPHER_AES_CBC))
如果(CPA_BITMAP_BIT_TESTCPA_CY_SYM_CIPHER_AES_CBC
{
{

    // algo is supported

}
}
else
其他
{
{


}
}
```

//支持算法

```
// algo is not supported
//不支持算法
```

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

客户端必须为此结构和任何需要内存的成员分配内存。当结构被传递给函数时，内存的所有权就传递给了函数。当函数返回时，内存的所有权返回给客户端。

---

## 8.13 Enumeration Type Documentation

## 8.14 枚举类型文档

enum **_CpaCySymPacketType**

列举型别_CpaCySymPacketTyp

Packet type for the cpaCySymPerformOp function

cpaCySymPerformOp 函数的数据包类型

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

枚举，用于向对称加密执行函数指示需要对哪种类型的数据包调用操作。当需要对客户端可获得的多个部分的消息执行处理时(例如，由于数据包的网络碎片)，多部分密码和散列操作非常有用。

**Note:**

**注意：**

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

对于支持部分数据包处理的操作有一些限制。有关详细信息，请参见函数 cpaCySymPerformOp

**See also:**
**另请参见：**

**cpaCySymPerformOp()**
cpaCySymPerformOp()

**Enumerator:**
**枚举器：**

| | |
|---|---|
| *CPA_CY_SYM_PACKET_TYPE_FULL* | Perform an operation on a full packet<br>*CPA _ CY _ SYM _数据包类型_完整对完整数据包执行操作* |
| *CPA_CY_SYM_PACKET_TYPE_PARTIAL* | Perform a partial operation and maintain the state of the partial operation within the session. This is used for either the first or subsequent packets within a partial packet flow.<br>*CPA_CY_SYM_PACKET_TYPE_PARTIAL 执行部分操作并维护*<br>会话中部分操作的状态。这用于部分分组流中的第一个或后续分组。 |
| *CPA_CY_SYM_PACKET_TYPE_LAST_PARTIAL* | Complete the last part of a multi-part operation<br>*CPA _ CY _ SYM _ PACKET _ TYPE _ LAST _ PARTIAL 完成多部分操作的最后一部分* |

enum **_CpaCySymOp**
列举型别_CpaCySymO

Types of operations supported by the cpaCySymPerformOp function.
cpaCySymPerformOp 函数支持的操作类型。

This enumeration lists different types of operations supported by the cpaCySymPerformOp function. The operation type is defined during session registration and cannot be changed for a session once it has been setup.
此枚举列出了由 cpaCySymPerformOp 函数支持的不同类型的操作。操作类型是在会话注册期间定义的，一旦设置，就不能为会话更改。

**See also:**
**另请参见：**

**cpaCySymPerformOp**
cpaCySymPerformOp

**Enumerator:**
**枚举器：**

| | |
|---|---|
| *CPA_CY_SYM_OP_NONE* | No operation<br>*CPA_CY_SYM_OP_NONE 无操作* |
| *CPA_CY_SYM_OP_CIPHER* | Cipher only operation on the data<br>*CPA_CY_SYM_OP_CIPHER 仅对数据进行加密操作* |
| *CPA_CY_SYM_OP_HASH* | Hash only operation on the data<br>*CPA_CY_SYM_OP_HASH 仅对数据进行哈希运算* |
| *CPA_CY_SYM_OP_ALGORITHM_CHAINING* | Chain any cipher with any hash operation. The |

*CPA _ CY _ SYM _ OP _ ALGORITHM _ CHAINING用任何哈希运算链接任何密码。这*
order depends on the value in the CpaCySymAlgChainOrder enum.
顺序取决于 CpaCySymAlgChainOrder 枚举中的值。

This value is also used for authenticated ciphers (GCM and CCM), in which case the cipherAlgorithm should take one of the values **CPA_CY_SYM_CIPHER_AES_CCM** or
这个值也用于认证密码(GCM 和 CCM)，在这种情况下，密码算法应该取其中一个值 CPA_CY_SYM_CIPHER_AES_CCM
**CPA_CY_SYM_CIPHER_AES_GCM**, while the hashAlgorithm should take the corresponding value **CPA_CY_SYM_HASH_AES_CCM** or **CPA_CY_SYM_HASH_AES_GCM**.
CPA_CY_SYM_CIPHER_AES_GCM，而 hashAlgorithm 应该取相应的值 CPA_CY_SYM_HASH_AES_CCM CPA_CY_SYM_HASH_AES_GCM

## enum _CpaCySymCipherAlgorithm
列举型别_CpaCySymCipherAlgorith

Cipher algorithms.
密码算法。

This enumeration lists supported cipher algorithms and modes.
此枚举列出了支持的密码算法和模式。

**Enumerator:**
**枚举器:**

| | |
|---|---|
| *CPA_CY_SYM_CIPHER_NULL* | NULL cipher algorithm. No mode applies to the NULL algorithm. |
| *SYM密码算法。没有模式适用于空算法。* | |
| *CPA_CY_SYM_CIPHER_ARC4* | (A)RC4 cipher algorithm |
| *CPA_CY_SYM_CIPHER_AES_ECB* | AES algorithm in ECB mode |
| *CPA_CY_SYM_CIPHER_AES_CBC* | AES algorithm in CBC mode |
| *CPA_CY_SYM_CIPHER_AES_CTR* | AES algorithm in Counter mode |
| *CPA_CY_SYM_CIPHER_AES_CCM* | |
| *CPA_CY_SYM_CIPHER_ARC4 (A)RC4密码算法* | *CPA_CY_SYM_CIPHER_AES_ECB* |
| | ECB 模式下的 AES 算法 |
| CPA_CY_SYM_CIPHER_AES_CBC | CBC 模式下的 AES 算法 CPA _ CY |
| _ SYM _密码_AES_CTR | 计数器模式下的 AES 算法 |
| CPA_CY_SYM_CIPHER_AES_CCM | |

AES algorithm in CCM mode. This authenticated cipher is only supported when the hash mode is also set to CPA_CY_SYM_HASH_MODE_AUTH. When this cipher algorithm is used the CPA_CY_SYM_HASH_AES_CCM element of the CpaCySymHashAlgorithm enum MUST be used to set up the related CpaCySymHashSetupData structure in the session context.

CCM 模式下的 AES 算法。仅当哈希模式也设置为 CPA _ CY _ SYM _哈希_模式_身份验证时，才支持此身份验证密码。使用此密码算法时，必须使用 CpaCySymHashAlgorithm 枚举的 CPA_CY_SYM_HASH_AES_CCM 元素在会话上下文中设置相关的 CpaCySymHashSetupData 结构。

*CPA_CY_SYM_CIPHER_AES_GCM* AES algorithm in GCM mode. This authenticated

*GCM 模式下 CPA_CY_SYM_CIPHER_AES_GCM AES 算法。这是经过验证的*

cipher is only supported when the hash mode is also set to CPA_CY_SYM_HASH_MODE_AUTH. When this cipher algorithm is used the CPA_CY_SYM_HASH_AES_GCM element of the CpaCySymHashAlgorithm enum MUST be used to set up the related CpaCySymHashSetupData structure in the session context.

仅当哈希模式也设置为 CPA _ CY _ SYM _哈希_模式_身份验证时，才支持密码。使用此密码算法时，必须使用 CpaCySymHashAlgorithm 枚举的 CPA_CY_SYM_HASH_AES_GCM 元素在会话上下文中设置相关的 CpaCySymHashSetupData 结构。

*CPA_CY_SYM_CIPHER_DES_ECB* DES algorithm in ECB mode

*CPA_CY_SYM_CIPHER_DES_CBC* DES algorithm in CBC mode

*CPA_CY_SYM_CIPHER_3DES_ECB* Triple DES algorithm in ECB mode

*CPA_CY_SYM_CIPHER_3DES_CBC* Triple DES algorithm in CBC mode

*CPA_CY_SYM_CIPHER_3DES_CTR* Triple DES algorithm in CTR mode

*CPA_CY_SYM_CIPHER_KASUMI_F8* Kasumi algorithm in F8 mode

*CPA_CY_SYM_CIPHER_SNOW3G_UEA2* SNOW3G algorithm in UEA2 mode

*CPA_CY_SYM_CIPHER_AES_F8* AES algorithm in F8 mode

*CPA_CY_SYM_CIPHER_AES_XTS* AES algorithm in XTS mode

*CPA_CY_SYM_CIPHER_ZUC_EEA3* ZUC algorithm in EEA3 mode

*CPA_CY_SYM_CIPHER_CHACHA* ChaCha20 Cipher Algorithm. This cipher is only

*ECB 模式下的 CPA_CY_SYM_CIPHER_DES_ECB DES 算法 CPA_CY_SYM_CIPHER_DES_CBC* CBC 模式下的 DES 算法 CPA_CY_SYM_CIPHER_3DES_ECB ECB 模式下的三重 DES 算法

CPA_CY_SYM_CIPHER_3DES_CBC CBC 模式下的三重 DES 算法

CPA_CY_SYM_CIPHER_3DES_CTR CTR 模式下的三重 DES 算法

CPA_CY_SYM_CIPHER_KASUMI_F8 F8 模式下的小霞算法

CPA_CY_SYM_CIPHER_SNOW3G_UEA2 模式下的 SNOW3G 算法 CPA_CY_SYM_CIPHER_AES_F8 F8 模式下的 AES 算法 CPA_CY_SYM_CIPHER_AES_XTS XTS 模式下的 AES 算法

CPA_CY_SYM_CIPHER_ZUC_EEA3 EEA3 模式下的 ZUC 算法

CPA_CY_SYM_CIPHER_CHACHA ChaCha20 密码算法。这个密码只有

supported for algorithm chaining. When selected, the 支持算法链接。选中时

hash algorithm must be set to CPA_CY_SYM_HASH_POLY and the hash mode must be set to CPA_CY_SYM_HASH_MODE_AUTH.

哈希算法必须设置为 CPA _ CY _ SYM _哈希_POLY，哈希模式必须设置为 CPA _ CY _ SYM _哈希_模式_验

证。

*CPA_CY_SYM_CIPHER_SM4_ECB* SM4 algorithm in ECB mode This cipher supports 128 bit keys only and does not support partial processing.
*CPA_CY_SYM_CIPHER_SM4_ECB SM4 算法在 ECB 模式下该密码支持 128* 仅位密钥，不支持部分处理。

*CPA_CY_SYM_CIPHER_SM4_CBC* SM4 algorithm in CBC mode This cipher supports 128 bit keys only and does not support partial processing.
*CPA_CY_SYM_CIPHER_SM4_CBC SM4 算法在 CBC 模式下该密码支持 128* 仅位密钥，不支持部分处理。

*CPA_CY_SYM_CIPHER_SM4_CTR* SM4 algorithm in CTR mode This cipher supports 128 bit keys only and does not support partial processing.
*CPA_CY_SYM_CIPHER_SM4_CTR SM4 算法在 CTR 模式下该密码支持 128* 仅位密钥，不支持部分处理。

## enum _CpaCySymCipherDirection

**列举型别_CpaCySymCipherDirectio**

Symmetric Cipher Direction
对称密码方向

This enum indicates the cipher direction (encryption or decryption).
此枚举指示密码方向(加密或解密)。

**Enumerator:**
**枚举器:**

*CPA_CY_SYM_CIPHER_DIRECTION_ENCRYPT* Encrypt Data
*CPA _ CY _ SYM _密码_方向_加密加密* 数据

*CPA_CY_SYM_CIPHER_DIRECTION_DECRYPT* Decrypt Data
*CPA _ CY _ SYM _密码_方向_解密解密* 数据

## enum _CpaCySymHashMode

**列举型别_CpaCySymHashMod**

Symmetric Hash mode
对称散列模式

This enum indicates the Hash Mode.
此枚举指示哈希模式。

**Enumerator:**

**枚举器:**

*CPA_CY_SYM_HASH_MODE_PLAIN* Plain hash. Can be specified for MD5 and the SHA
*CPA _ CY _ SYM _哈希_模式_纯哈希。可以为 MD5 和 SHA 指定*
family of hash algorithms.
哈希算法家族。

*CPA_CY_SYM_HASH_MODE_AUTH* Authenticated hash. This mode may be used in
*CPA _ CY _ SYM _哈希_模式_认证哈希。该模式可用于*
conjunction with the MD5 and SHA family of algorithms to specify HMAC. It MUST also be specified with all of the remaining algorithms, all of which are in fact authentication algorithms.
与 MD5 和 SHA 系列算法一起指定 HMAC。还必须用所有剩余的算法来指定它，所有这些算法实际上都是认证算法。

*CPA_CY_SYM_HASH_MODE_NESTED* Nested hash. Can be specified for MD5 and the SHA
*CPA _ CY _ SYM _哈希_模式_嵌套嵌套哈希。可以为 MD5 和 SHA 指定*
family of hash algorithms.
哈希算法家族。

enum **_CpaCySymHashAlgorithm**
**列举型别_CpaCySymHashAlgorith**

Hash algorithms.
哈希算法。

This enumeration lists supported hash algorithms.
此枚举列出了支持的哈希算法。

**Enumerator:**
**枚举器:**

*CPA_CY_SYM_HASH_NONE* No hash algorithm.
*CPA _ CY _ SYM _哈希_无无哈希算法。*
*CPA_CY_SYM_HASH_MD5* MD5 algorithm. Supported in all 3 hash modes
*CPA_CY_SYM_HASH_SHA1* 128 bit SHA algorithm. Supported in all 3 hash modes
*CPA_CY_SYM_HASH_SHA224* 224 bit SHA algorithm. Supported in all 3 hash modes
*CPA_CY_SYM_HASH_SHA256* 256 bit SHA algorithm. Supported in all 3 hash modes
*CPA_CY_SYM_HASH_SHA384* 384 bit SHA algorithm. Supported in all 3 hash modes
*CPA_CY_SYM_HASH_SHA512* 512 bit SHA algorithm. Supported in all 3 hash modes
*CPA_CY_SYM_HASH_AES_XCBC* AES XCBC algorithm. This is only supported in the hash
*CPA _ CY _ SYM _哈希_MD5 MD5 算法。所有 3 种哈希模式都支持 CPA _ CY _ SYM _哈希_SHA1*
128 位 SHA 算法。所有 3 种哈希模式都支持 CPA _ CY _
SYM _哈希_SHA224 224 位 SHA 算法。所有 3 种哈希模式都支持 CPA _ CY _
SYM _哈希_SHA256 256 位 SHA 算法。所有 3 种哈希模式都支持 CPA _ CY _
SYM _哈希_SHA384 384 位 SHA 算法。所有 3 种哈希模式都支持 CPA _ CY _
SYM _哈希_SHA512 512 位 SHA 算法。所有 3 种哈希模式都支持 CPA _ CY _
SYM _哈希_AES_XCBC AES XCBC 算法。这仅在哈希中受支持
mode CPA_CY_SYM_HASH_MODE_AUTH.
模式 CPA _ CY _ SYM _哈希_模式_认证。

*CPA_CY_SYM_HASH_AES_CCM* AES algorithm in CCM mode. This authenticated cipher
*CCM 模式下的 CPA _ CY _ SYM _哈希_AES_CCM AES 算法。这个经过验证的密码*
requires that the hash mode is set to

CPA_CY_SYM_HASH_MODE_AUTH. When this hash algorithm is used, the CPA_CY_SYM_CIPHER_AES_CCM element of the CpaCySymCipherAlgorithm enum MUST be used to set up the related CpaCySymCipherSetupData structure in the session context.
要求将哈希模式设置为 CPA _ CY _ SYM _哈希_模式_验证。使用此哈希算法时，必须使用 CpaCySymCipherAlgorithm 枚举的 CPA_CY_SYM_CIPHER_AES_CCM 元素在会话上下文中设置相关的 CpaCySymCipherSetupData 结构。

*CPA_CY_SYM_HASH_AES_GCM* AES algorithm in GCM mode. This authenticated cipher
*GCM 模式下的 CPA_CY_SYM_HASH_AES_GCM AES 算法。这个经过验证的密码*
requires that the hash mode is set to CPA_CY_SYM_HASH_MODE_AUTH. When this hash algorithm is used, the CPA_CY_SYM_CIPHER_AES_GCM element of the CpaCySymCipherAlgorithm enum MUST be used to set up the related CpaCySymCipherSetupData structure in the session context.
要求将哈希模式设置为 CPA _ CY _ SYM _哈希_模式_验证。使用此哈希算法时，必须使用 CpaCySymCipherAlgorithm 枚举的 CPA_CY_SYM_CIPHER_AES_GCM 元素在会话上下文中设置相关的 CpaCySymCipherSetupData 结构。

*CPA_CY_SYM_HASH_KASUMI_F9* Kasumi algorithm in F9 mode. This is only supported in
*F9 模式下的 CPA _ CY _ SYM _哈希_KASUMI_F9 小霞算法。这仅在中受支持*
the hash mode CPA_CY_SYM_HASH_MODE_AUTH.
哈希模式 CPA _ CY _ SYM _哈希_MODE_AUTH。

*CPA_CY_SYM_HASH_SNOW3G_UIA2* SNOW3G algorithm in UIA2 mode. This is only
*UIA2 模式下的 CPA _ CY _ SYM _ HASH _ snow 3g _ ui a2 snow 3g 算法。这只是*
supported in the hash mode CPA_CY_SYM_HASH_MODE_AUTH.
在哈希模式 CPA _ CY _ SYM _哈希 _MODE_AUTH 中支持。

*CPA_CY_SYM_HASH_AES_CMAC* AES CMAC algorithm. This is only supported in the hash
*CPA _ CY _ SYM _哈希_AES_CMAC AES CMAC 算法。这仅在哈希中受支持*
mode CPA_CY_SYM_HASH_MODE_AUTH.
模式 CPA _ CY _ SYM _哈希_模式_认证。

*CPA_CY_SYM_HASH_AES_GMAC* AES GMAC algorithm. This is only supported in the hash
*CPA _ CY _ SYM _哈希_AES_GMAC AES GMAC 算法。这仅在哈希中受支持*
mode CPA_CY_SYM_HASH_MODE_AUTH. When this hash algorithm is used, the CPA_CY_SYM_CIPHER_AES_GCM element of the CpaCySymCipherAlgorithm enum MUST be used to set
模式 CPA _ CY _ SYM _哈希_模式_认证。使用此哈希算法时，必须使用 CpaCySymCipherAlgorithm 枚举的 CPA_CY_SYM_CIPHER_AES_GCM 元素来设置

up the related CpaCySymCipherSetupData structure in the session context.

在会话上下文中打开相关的 CpaCySymCipherSetupData 结构。

*CPA_CY_SYM_HASH_AES_CBC_MAC* AES-CBC-MAC algorithm. This is only supported in the hash mode CPA_CY_SYM_HASH_MODE_AUTH. Only 128-bit keys are supported.

*CPA _ CY _ SYM _哈希_AES_CBC_MAC AES-CBC-MAC 算法。这仅在中受支持*哈希模式 CPA _ CY _ SYM _哈希_模式_认证。仅支持 128 位密钥。

*CPA_CY_SYM_HASH_ZUC_EIA3* ZUC algorithm in EIA3 mode

*EIA3 模式下的 CPA _ CY _ SYM _哈希_ZUC_EIA3 ZUC 算法*

*CPA_CY_SYM_HASH_SHA3_256* 256 bit SHA-3 algorithm. Only CPA_CY_SYM_HASH_MODE_PLAIN and CPA_CY_SYM_HASH_MODE_AUTH are supported, that is, the hash mode CPA_CY_SYM_HASH_MODE_NESTED is not supported for this algorithm. Partial requests are not supported, that is, only requests of CPA_CY_SYM_PACKET_TYPE_FULL are supported.

*CPA _ CY _ SYM _哈希_SHA3_256 256 位 SHA-3 算法。仅仅*支持 CPA_CY_SYM_HASH_MODE_PLAIN 和 CPA_CY_SYM_HASH_MODE_AUTH，即该算法不支持 CPA_CY_SYM_HASH_MODE_NESTED 哈希模式。不支持部分请求，即只支持 CPA_CY_SYM_PACKET_TYPE_FULL 的请求。

*CPA_CY_SYM_HASH_SHA3_224* 224 bit SHA-3 algorithm. Only CPA_CY_SYM_HASH_MODE_PLAIN and CPA_CY_SYM_HASH_MODE_AUTH are supported, that is, the hash mode CPA_CY_SYM_HASH_MODE_NESTED is not supported for this algorithm.

*CPA _ CY _ SYM _哈希_SHA3_224 224 位 SHA-3 算法。仅仅*支持 CPA_CY_SYM_HASH_MODE_PLAIN 和 CPA_CY_SYM_HASH_MODE_AUTH，即该算法不支持 CPA_CY_SYM_HASH_MODE_NESTED 哈希模式。

*CPA_CY_SYM_HASH_SHA3_384* 384 bit SHA-3 algorithm. Only CPA_CY_SYM_HASH_MODE_PLAIN and CPA_CY_SYM_HASH_MODE_AUTH are supported, that is, the hash mode CPA_CY_SYM_HASH_MODE_NESTED is not supported for this algorithm. Partial requests are not supported, that is, only requests of CPA_CY_SYM_PACKET_TYPE_FULL are supported.

*CPA _ CY _ SYM _哈希_SHA3_384 384 位 SHA-3 算法。仅仅*支持 CPA_CY_SYM_HASH_MODE_PLAIN 和 CPA_CY_SYM_HASH_MODE_AUTH，即该算法不支持 CPA_CY_SYM_HASH_MODE_NESTED 哈希模式。不支持部分请求，即只支持 CPA_CY_SYM_PACKET_TYPE_FULL 的请求。

*CPA_CY_SYM_HASH_SHA3_512* 512 bit SHA-3 algorithm. Only CPA_CY_SYM_HASH_MODE_PLAIN and CPA_CY_SYM_HASH_MODE_AUTH are supported, that is, the hash mode

*CPA _ CY _ SYM _哈希_SHA3_512 512 位 SHA-3 算法。仅仅*支持 CPA_CY_SYM_HASH_MODE_PLAIN 和 CPA_CY_SYM_HASH_MODE_AUTH，即该算法不支持 CPA_CY_SYM_HASH_MODE_NESTED 哈希模式。不支持部分请求，即只支持 CPA_CY_SYM_PACKET_TYPE_FULL 的请求。

CPA_CY_SYM_HASH_MODE_NESTED is not supported for this algorithm. Partial requests are not supported, that is, only requests of CPA_CY_SYM_PACKET_TYPE_FULL are supported.
支持 CPA_CY_SYM_HASH_MODE_AUTH，即该算法不支持 CPA_CY_SYM_HASH_MODE_NESTED 哈希模式。不支持部分请求，即只支持 CPA_CY_SYM_PACKET_TYPE_FULL 的请求。

*CPA_CY_SYM_HASH_SHAKE_128* 128 bit SHAKE algorithm. This is only supported in the hash mode CPA_CY_SYM_HASH_MODE_PLAIN. Partial requests are not supported, that is, only requests of CPA_CY_SYM_PACKET_TYPE_FULL are supported.
*CPA_CY_SYM_HASH_SHAKE_128 128 位摇动算法。这仅在中受支持*
哈希模式 CPA _ CY _ SYM _哈希_模式_普通。不支持部分请求，即只支持 CPA_CY_SYM_PACKET_TYPE_FULL 的请求。

*CPA_CY_SYM_HASH_SHAKE_256* 256 bit SHAKE algorithm. This is only supported in the hash mode CPA_CY_SYM_HASH_MODE_PLAIN. Partial requests are not supported, that is, only requests of CPA_CY_SYM_PACKET_TYPE_FULL are supported.
*CPA_CY_SYM_HASH_SHAKE_256 256 位摇动算法。这仅在中受支持*
哈希模式 CPA _ CY _ SYM _哈希_模式_普通。不支持部分请求，即只支持 CPA_CY_SYM_PACKET_TYPE_FULL 的请求。

*CPA_CY_SYM_HASH_POLY* Poly1305 hash algorithm. This is only supported in the hash mode CPA_CY_SYM_HASH_MODE_AUTH. This hash algorithm is only supported as part of an algorithm chain with AES_CY_SYM_CIPHER_CHACHA to implement the ChaCha20-Poly1305 AEAD algorithm.
*CPA _ CY _ SYM _哈希_POLY Poly1305 哈希算法。这仅在中受支持*
哈希模式 CPA _ CY _ SYM _哈希_模式_认证。此哈希算法仅支持作为 AES_CY_SYM_CIPHER_CHACHA 算法链的一部分来实现 ChaCha20-Poly1305 AEAD 算法。

*CPA_CY_SYM_HASH_SM3* SM3 hash algorithm. Supported in all 3 hash modes.
*CPA _ CY _ SYM _哈希_SM3 SM3 哈希算法。所有 3 种哈希模式都支持。*

enum **_CpaCySymAlgChainOrder**
列举型别_CpaCySymAlgChainOrde

Algorithm Chaining Operation Ordering

算法链接操作排序

This enum defines the ordering of operations for algorithm chaining.
此枚举定义算法链的操作顺序。

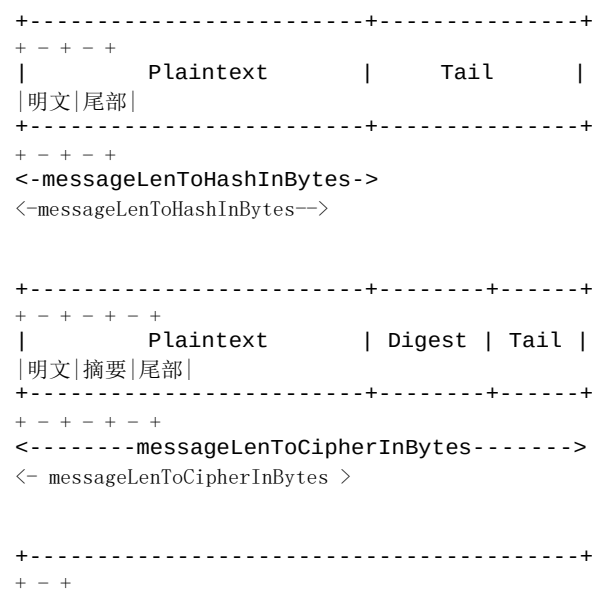
**Enumerator:**
**枚举器:**

*CPA_CY_SYM_ALG_CHAIN_ORDER_HASH_THEN_CIPHER*
*CPA _ CY _ SYM _算法_链_顺序_散列_ THEN _密码*

Perform the hash operation followed by the cipher operation. If it is required that the result of the hash (i.e. the digest) is going to be included in the data to be ciphered, then:
执行哈希运算，然后执行密码运算。如果需要将散列的结果(即摘要)包含在要加密的数据中，则:

- ◊ The digest MUST be placed in the destination buffer at the location corresponding to the end of the data region to be hashed (hashStartSrcOffsetInBytes + messageLenToHashInBytes), i.e. there must be no gaps between the start of the digest and the end of the data region to be hashed.
- ◊ 摘要必须放在目标缓冲区中与要哈希的数据区域的结尾相对应的位置(hashStartSrcOffsetInBytes+messageLenToHashInBytes)，即在摘要的开头和要哈希的数据区域的结尾之间必须没有间隙。
- ◊ The messageLenToCipherInBytes member of the CpaCySymOpData structure must be equal to the overall length of the plain text, the digest length and any (optional) trailing data that is to be included.
- ◊ CpaCySymOpData 结构的 messageLenToCipherInBytes 成员必须等于纯文本的总长度、摘要长度和要包含的任何(可选)尾部数据。
- ◊ The messageLenToCipherInBytes must be a multiple to the block size if a block cipher is
- ◊ 如果块密码为，则 messageLenToCipherInBytes 必须是块大小的倍数
  being used.
  被利用了。

The following is an example of the layout of the buffer before the operation, after the hash, and after the cipher:
以下是运算前、哈希后和加密后的缓冲区布局示例:



```
|                Cipher Text                |
密码文本|
+-------------------------------------------+
+ - +
```

*CPA_CY_SYM_ALG_CHAIN_ORDER_CIPHER_THEN_HASH*
*CPA _ CY _ SYM _ ALG _链_顺序_密码_ THEN _哈希*
Perform the cipher operation followed by the hash operation. The hash operation will be performed on the ciphertext resulting from the cipher operation.
执行密码运算，然后执行哈希运算。哈希运算将在密码运算产生的密文上执行。

The following is an example of the layout of the buffer before the operation, after the cipher, and after the hash:
以下是运算前、加密后和哈希后的缓冲区布局示例：

```
+--------+---------------------------+---------------+
+ - + - + - +
|  Head  |         Plaintext         |    Tail       |
|头|明文|尾|
+--------+---------------------------+---------------+
+ - + - + - +
         <-messageLenToCipherInBytes->
         <-messageLenToCipherInBytes-->
```

```
+--------+---------------------------+---------------+
+ - + - + - +
|  Head  |         Ciphertext        |    Tail       |
|头|密文|尾|
+--------+---------------------------+---------------+
+ - + - + - +
<------messageLenToHashInBytes------>
<- messageLenToHashInBytes >
```

```
+--------+---------------------------+--------+------+
+ - + - + - + - +
|  Head  |         Ciphertext        | Digest | Tail |
|头|密文|摘要|尾
+--------+---------------------------+--------+------+
+ - + - + - + - +
```

## 8.15 Function Documentation

## 8.16 功能文档

```
                                                                    instanceHandle,
cpaCySymSessionCtxGetSize          ( const
                                     const * pSessionSetupData,
                                     *pSessionCtxSizeInBytes
                                   )
```

Gets the size required to store a session context.
获取存储会话上下文所需的大小。

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

For a given implementation of this API, it is safe to assume that **cpaCySymSessionCtxGetSize()** will always return the same size and that the size will not be different for different setup data parameters. However, it should be noted that the size may change: (1) between different implementations of the API (e.g. between software and hardware implementations or between different hardware implementations) (2) between different releases of the same API implementation.
对于该 API 的给定实现，可以安全地假设 cpaCySymSessionCtxGetSize()

The size returned by this function is the smallest size needed to support all possible combinations of setup data parameters. Some setup data parameter combinations may fit within a smaller session context size. The alternate **cpaCySymSessionCtxGetDynamicSize()** function will return the smallest size needed to fit the provided setup data parameters.
此函数返回的大小是支持设置数据参数的所有可能组合所需的最小大小。一些设置数据参数组合可能适合较小的会话上下文大小。替补队员 cpaCySymSessionCtxGetDynamicSize()

**Context:**
**背景：**

This is a synchronous function that cannot sleep. It can be executed in a context that does not permit sleeping.
这是一个不能睡眠的同步功能。它可以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.<br>[in] instanceHandle 执行个体控制代码。 |
| [in] | *pSessionSetupData* | Pointer to session setup data which contains parameters which are static for a given cryptographic session such as operation type, mechanisms, and keys for cipher and/or hash operations.<br>[in]指向会话设置数据的 pSessionSetupData 指针，该数据包含对于给定加密会话是静态的，例如加密和/或散列操作的操作类型、机制和密钥。 |
| [out] | *pSessionCtxSizeInBytes* | The amount of memory in bytes required to hold the Session Context.<br>[out]psessiontxsizeinbytes 保存会话所需的内存量(以字节为单位)<br>语境。 |

**Return values:**

**返回值:**

| | |
|---|---|
| *CPA_STATUS_SUCCESS* | Function executed successfully.<br>*CPA_STATUS_SUCCESS 函数执行成功。* |
| *CPA_STATUS_FAIL* | Function failed.<br>*CPA_STATUS_FAIL 函数失败。* |

*CPA_STATUS_INVALID_PARAM* Invalid parameter passed in.
*CPA_STATUS_RESOURCE* Error related to system resources.
*CPA_STATUS_UNSUPPORTED* Function is not supported.

*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 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:**

This is a synchronous function and has no completion callback associated with it.
这是一个同步函数，没有与之关联的完成回调。

**See also:**

**另请参见:**

**CpaCySymSessionSetupData cpaCySymInitSession() cpaCySymSessionCtxGetDynamicSize() cpaCySymPerformOp()**
CpaCySymSessionSetupData cpaCySymInitSession() cpaCySymSessionCtxGetDynamicSize() cpaCySymPerformOp()

**CpaStatus**
CpaStatus
cpaCySymSessionCtxGetDynamicSize ( const **CpaInstanceHandle**
cpacysymsessiontxgetdynamicsize(const**CpaInstanceHandle**
const
常数
**CpaCySymSessionSetupData** *
CpaCySymSessionSetupData *

*instanceHandle*, *pS*

*essionSetupData*,

*instanceHandle*, *pSessionSetupData*,

**Cpa32U** * *pSessionCtxSizeInBytes*

Cpa32U * pSessionCtxSizeInBytes

)

)

Gets the minimum size required to store a session context.
获取存储会话上下文所需的最小大小。

This function is used by the client to determine the smallest size of the memory it must allocate in order to store the session context. This MUST be called before the client allocates the memory for the session context and before the client calls the **cpaCySymInitSession** function.
客户端使用此函数来确定它必须分配的最小内存大小，以便存储会话上下文。在客户端为会话上下文分配内存之前，以及在客户端调用 cpaCySymInitSession

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

**Context:**
**背景:**
This is a synchronous function that cannot sleep. It can be executed in a context that does not permit sleeping.
这是一个不能睡眠的同步功能。它可以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] *pSessionSetupData* Pointer to session setup data which contains parameters which are static for a given cryptographic session such as operation type, mechanisms, and keys for cipher and/or hash operations.
[in]指向会话设置数据的 pSessionSetupData 指针，该数据包含对于给定加密会话是静态的，例如加密和/或散列操作的操作类型、机制和密钥。

[out] *pSessionCtxSizeInBytes* The amount of memory in bytes required to hold the Session Context.
[out]psessiontxsizeinbytes 保存会话所需的内存量(以字节为单位)语境。

**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 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:**

This is a synchronous function and has no completion callback associated with it.
这是一个同步函数，没有与之关联的完成回调。

**See also:**

**另请参见:**

**CpaCySymSessionSetupData cpaCySymInitSession() cpaCySymSessionCtxGetSize() cpaCySymPerformOp()**

CpaCySymSessionSetupData cpaCySymInitSession()
cpaCySymSessionCtxGetSize()cpaCySymPerformOp()

```
cpaCySymInitSession (  const                                    instanceHandle
cpaCySymInitSession (  const                                    instanceHandle,
                            const pSymCb,
                            const *  pSessionSetupData,
                            sessionCtx
                          )
```

8.13 Function

Initialize a session for symmetric cryptographic API.
初始化对称加密 API 的会话。

This function is used by the client to initialize an asynchronous completion callback function for the symmetric cryptographic operations. Clients MAY register multiple callback functions using this function. The callback function is identified by the combination of userContext, pSymCb and session context (sessionCtx). The session context is the handle to the session and needs to be passed when processing calls. Callbacks on completion of operations within a session are guaranteed to be in the same order they were submitted in.
客户端使用此函数来初始化对称加密操作的异步完成回调函数。客户端可以使用这个函数注册多个回调函数。回调函数由 userContext、pSymCb 和会话上下文(sessionCtx)的组合来标识。会话上下文是会话的句柄，需要在处理调用时传递。会话中操作完成时的回调保证与提交时的顺序相同。

**Context:**
**背景：**

This is a synchronous function and it cannot sleep. It can be executed in a context that does not permit sleeping.
这是一个同步功能，它不能休眠。它可以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] *pSymCb* Pointer to callback function to be registered. Set to NULL if the
[in]指向要注册的回调函数的 pSymCb 指针。如果
cpaCySymPerformOp function is required to work in a synchronous manner.
cpacysymperformatopfunction 需要以同步方式工作。

[in] *pSessionSetupData* Pointer to session setup data which contains parameters which are
[in]指向会话设置数据的 pSessionSetupData 指针，该数据包含以下参数
static for a given cryptographic session such as operation type, mechanisms, and keys for cipher and/or hash operations.
给定加密会话的静态信息，例如加密和/或哈希操作的操作类型、机制和密钥。

[out] *sessionCtx* Pointer to the memory allocated by the client to store the session
[out] sessionCtx 指向客户端为存储会话而分配的内存的指针
context. This will be initialized with this function. This value needs to be passed to subsequent processing calls.
语境。这将用这个函数来初始化。这个值需要传递给后续的处理调用。

**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 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：**

This is a synchronous function and has no completion callback associated with it.
这是一个同步函数，没有与之关联的完成回调。

**See also:**

**另请参见：**

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

CpaCySymSessionCtx, CpaCySymCbFuncCpaCySymSessionSetupDatacpaCySymRemoveSession()cpaCySymPerformOp()

Remove (delete) a symmetric cryptographic session.

**CpaStatus** cpaCySymRemoveSession ( const **CpaInstanceHandle** *instanceHandle*, **CpaCySymSessionCtx** *pSessionCtx*

CpaStatus cpaCySymRemoveSession( const CpaInstanceHandle instanceHandle, CpaCySymSessionCtx pSessionCtx

移除（删除）对称加密会话。

This function will remove a previously initialized session context and the installed callback handler function. Removal will fail if outstanding calls still exist for the initialized session handle. The client needs to retry the remove function at a later time. The memory for the session context MUST not be freed until this call has completed successfully.
此函数将删除先前初始化的会话上下文和已安装的回调处理函数。如果初始化的会话句柄仍存在未完成的调用，移除将会失败。客户端需要稍后重试删除功能。在此调用成功完成之前，不得释放会话上下文的内存。

**Context:**

**背景：**

This is a synchronous function that cannot sleep. It can be executed in a context that does not permit sleeping.
这是一个不能睡眠的同步功能。它可以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,out] *pSessionCtx* Session context to be removed.
[in, out]要移除的 pSessionCtx 会话上下文。

**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:**

Note that this is a synchronous function and has no completion callback associated with it.
请注意，这是一个同步函数，没有与之关联的完成回调。

**See also:**

**另请参见:**

**CpaCySymSessionCtx**, **cpaCySymInitSession()**
CpaCySymSessionCtx, cpaCySymInitSession()

```
cpaCySymUpdateSession (                                    sessionCtx
cpaCySymUpdateSession  (                                   sessionCtx,
                                  const *  pSessionUpdateData
                                )
```

Update a session.
更新会话。

This function is used to update certain parameters of a session, as specified by the CpaCySymSessionUpdateData data structure.
此函数用于更新由 CpaCySymSessionUpdateData 数据结构指定的会话的某些参数。

It can be used on sessions created with either the so-called Traditional API (**cpaCySymInitSession**) or the Data Plane API (**cpaCySymDpInitSession**).
它可以用于通过所谓的传统 API(cpaCySymInitSessioncpaCySymDpInitSession

In order for this function to operate correctly, two criteria must be met:
为了使该功能正常运行，必须满足两个标准:

- In the case of sessions created with the Traditional API, the session must be stateless, i.e. the field partialsNotRequired of the CpaCySymSessionSetupData data structure must be FALSE. (Sessions created using the Data Plane API are always stateless.)
- 在使用传统 API 创建会话的情况下，会话必须是无状态的，即 CpaCySymSessionSetupData 数据结构的字段 partialsNotRequired 必须为 FALSE。(使用数据平面 API 创建的会话总是无状态的。)

- There must be no outstanding requests in flight for the session. The application can call the function
- 该会话必须没有未完成的请求。应用程序可以调用该函数
  **cpaCySymSessionInUse** to test for this.
  cpaCySymSessionInUse 来测试这个。

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

请注意，在多线程应用程序(仅使用传统 API 支持)的情况下，此函数可能会失败，即使之前调用了该函数 cpaCySymSessionInUse

**Parameters:**
**参数:**

[in] *sessionCtx* Identifies the session to be reset.
[in] sessionCtx 标识要重置的会话。
[in] *pSessionUpdateData* Pointer to session data which contains the parameters to be updated.
[in]指向包含要更新的参数的会话数据的 pSessionUpdateData 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:**

This is a synchronous function and has no completion callback associated with it.
这是一个同步函数，没有与之关联的完成回调。

Indicates whether there are outstanding requests on a given session.
指示给定会话中是否有未完成的请求。

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

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

### 2.12 Function

此函数用于测试指定会话中是否有未完成的请求。这可以在使用函数 cpaCySymResetSession 重置会话参数之前使用。请参阅关于该函数的多线程应用的一些附加说明。

**Parameters:**
**参数:**

[in] *sessionCtx* Identifies the session to be reset.
[in] sessionCtx 标识要重置的会话。
[out] *pSessionInUse* Returns CPA_TRUE if there are outstanding requests on the session, or CPA_FALSE otherwise.
[out]如果会话中有未完成的请求，pSessionInUse 将返回 CPA_TRUE，否则返回 CPA_FALSE。

```
const
const
cpaCySymPerformOp    (                                          instanceHandle, pCallbackTag, pOpData,
                                                                pSrcBuffer, pDstBuffer, pVerifyResult
void * const
* const * * *
)
```

Perform a symmetric cryptographic operation on an existing session.

对现有会话执行对称加密操作。

Performs a cipher, hash or combined (cipher and hash) operation on the source data buffer using supported symmetric key algorithms and modes.
使用支持的对称密钥算法和模式对源数据缓冲区执行加密、哈希或组合(加密和哈希)操作。

This function maintains cryptographic state between calls for partial cryptographic operations. If a partial cryptographic operation is being performed, then on a per-session basis, the next part of the multi-part message can be submitted prior to previous parts being completed, the only limitation being that all parts must be performed in sequential order.
该函数维护部分加密操作调用之间的加密状态。如果正在执行部分加密操作，则在每个会话的基础上，多部分消息的下一部分可以在前面部分完成之前提交，唯一的限制是所有部分必须按顺序执行。

If for any reason a client wishes to terminate the partial packet processing on the session (for example if a packet fragment was lost) then the client MUST remove the session.
如果出于任何原因，客户端希望终止会话上的部分分组处理(例如，如果分组片段丢失)，则客户端必须移除该会话。

When using partial packet processing with algorithm chaining, only the cipher state is maintained between calls. The hash state is not be maintained between calls. Instead the hash digest will be generated/verified for each call. If both the cipher state and hash state need to be maintained between calls, algorithm chaining cannot be used.
当使用带有算法链的部分数据包处理时，在调用之间仅维护密码状态。在调用之间不保持哈希状态。相反，将为每个调用生成/验证哈希摘要。如果在调用之间需要维护密码状态和散列状态，则不能使用算法链。

The following restrictions apply to the length:
以下限制适用于长度:

- When performing block based operations on a partial packet (excluding the final partial packet), the data that is to be operated on MUST be a multiple of the block size of the algorithm being used. This restriction only applies to the cipher state when using partial packets with algorithm chaining.
- 当对部分包(不包括最后的部分包)执行基于块的操作时，要操作的数据必须是所用算法的块大小的倍数。此限制仅适用于在算法链中使用部分数据包时的密码状态。

- The final block must not be of length zero (0) if the operation being performed is the authentication algorithm **CPA_CY_SYM_HASH_AES_XCBC**. This is because this algorithm requires that the final block be XORed with another value internally. If the length is zero, then the return code **CPA_STATUS_INVALID_PARAM** will be returned.
- 如果正在执行的操作是身份验证算法，则最后一个块的长度不能为零(0)CPA_CY_SYM_HASH_AES_XCBCCPA_STATUS_INVALID_PARAM

- The length of the final block must be greater than or equal to 16 bytes when using the
- 当使用时，最后一个块的长度必须大于或等于 16 字节
  **CPA_CY_SYM_CIPHER_AES_XTS** cipher algorithm.
  CPA_CY_SYM_CIPHER_AES_XTS 密码算法。

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

仅当下列条件成立时，才支持部分数据包处理：

- The cipher, hash or authentication operation is "in place" (that is, pDstBuffer == pSrcBuffer)
- 密码、哈希或身份验证操作“就位”（即 pDstBuffer == pSrcBuffer）

- The cipher or hash algorithm is NOT one of Kasumi or SNOW3G
- 密码或哈希算法不是小霞或 SNOW3G 中的一种

- The cipher mode is NOT F8 mode.
- 密码模式不是 F8 模式。

- The hash algorithm is NOT SHAKE
- 哈希算法没有动摇

- The cipher algorithm is not SM4
- 密码算法不是 SM4

- The cipher algorithm is not CPA_CY_SYM_CIPHER_CHACHA and the hash algorithm is not CPA_CY_SYM_HASH_POLY.
- 密码算法不是 CPA_CY_SYM_CIPHER_CHACHA，哈希算法不是 CPA _ CY _ SYM _哈希_POLY。

- The instance/implementation supports partial packets as one of its capabilities (see
- 实例/实现支持部分数据包作为其功能之一(参见
  **CpaCySymCapabilitiesInfo**).
  CpaCySymCapabilitiesInfo).

The term "in-place" means that the result of the cryptographic operation is written into the source buffer. The term "out-of-place" means that the result of the cryptographic operation is written into the destination buffer. To perform "in-place" processing, set the pDstBuffer parameter to point at the same location as the pSrcBuffer parameter.
术语“就地”意味着加密操作的结果被写入源缓冲器。术语“不在适当的位置”意味着加密操作的结果被写入目的缓冲器。若要执行“就地”处理，请将 pDstBuffer 参数设置为指向与 pSrcBuffer 参数相同的位置。

**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 data that will be returned to the client in the callback.
[in] pCallbackTag 将在回调中返回给客户端的不透明数据。

[in] *pOpData* Pointer to a structure containing request parameters. The client code allocates the memory for this structure. This component takes ownership of the memory until it is returned in the callback.
[in]指向包含请求参数的结构的 pOpData 指针。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。

[in] *pSrcBuffer* The source buffer. The caller MUST allocate the source buffer and populate it with data. For optimum performance, the data pointed to SHOULD be 8-byte aligned. For block ciphers, the data passed in MUST be a multiple of the relevant block size. i.e. padding WILL NOT be applied to the data. For optimum performance, the buffer should only contain the data region that the cryptographic operation(s) must be performed on. Any additional data in the source buffer may be copied to the destination buffer and this copy may degrade performance.

8.12 Function

[in] pSrcBuffer 来源缓冲区。调用者必须分配源缓冲区并用数据填充它。为了获得最佳性能，指向的数据应该 8 字节对齐。对于块密码，传入的数据必须是相关块大小的倍数。即填充将不会应用于数据。为了获得最佳性能，缓冲区应该只包含必须执行加密操作的数据区域。源缓冲区中的任何附加数据都可能被复制到目标缓冲区，这种复制可能会降低性能。

[out] *pDstBuffer* The destination buffer. The caller MUST allocate a sufficiently sized destination buffer to hold the data output (including the authentication tag in the case of CCM). Furthermore, the destination buffer must be the same size as the source buffer (i.e. the sum of lengths of the buffers in the buffer list must be the same). This effectively means that the source buffer must in fact be big enough to hold the output data, too. This is because, for out-of-place processing, the data outside the regions in the source buffer on which cryptographic operations are performed are copied into the destination buffer. To perform "in-place" processing set the pDstBuffer parameter in cpaCySymPerformOp function to point at the same location as pSrcBuffer. For optimum performance, the data pointed to SHOULD be 8-byte aligned.

[out] pDstBuffer 目的缓冲区。调用者必须分配足够大的目的缓冲区来保存数据输出(在 CCM 的情况下包括认证标签)。此外，目标缓冲区必须与源缓冲区大小相同(即缓冲区列表中缓冲区的长度总和必须相同)。这实际上意味着源缓冲区也必须足够大以容纳输出数据。这是因为，对于错位处理，源缓冲区中执行加密操作的区域之外的数据被复制到目标缓冲区中。若要执行“就地”处理，请将 cpacysymperformatopfunction 中的 pDstBuffer 参数设置为指向与 pSrcBuffer 相同的位置。为了获得最佳性能，指向的数据应该 8 字节对齐。

[out] *pVerifyResult* In synchronous mode, this parameter is returned when the verifyDigest option is set in the CpaCySymSessionSetupData structure. A value of CPA_TRUE indicates that the compare succeeded. A value of CPA_FALSE indicates that the compare failed for an unspecified reason.

[out] pVerifyResult 在同步模式下，如果在 CpaCySymSessionSetupData 结构中设置了 verifyDigest 选项，则返回此参数。CPA_TRUE 值表示比较成功。CPA_FALSE 值表示比较因未指明的原因而失败。

**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_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 resource. |
| *CPA_STATUS_RESTARTING* | API implementation is restarting. Resubmit the request. |
| *CPA_STATUS_UNSUPPORTED* | Function is not supported. |

*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. A Cryptographic session has been previously setup using the **cpaCySymInitSession** function call.
该组件已通过 cpaCyStartInstance 函数初始化。先前已经使用设置了加密会话 cpaCySymInitSession

**Postcondition:**

**后置条件:**

None
没有人

**Note:**

**注意:**

When in asynchronous

mode, a callback of type CpaCySymCbFunc is generated in response to this function call. Any errors generated during processing are reported as part of the callback status code.
在异步模式下，会生成一个类型为 CpaCySymCbFunc 的回调来响应此函数调用。处理过程中产生的任何错误都会作为回调状态代码的一部分进行报告。

**See also:**

**另请参见:**

**CpaCySymOpData**, **cpaCySymInitSession()**, **cpaCySymRemoveSession()**
CpaCySymOpData, cpaCySymInitSession() cpaCySymRemoveSession()

**CpaStatus CPA_DEPRECATED** cpaCySymQueryStats ( const **CpaInstanceHandle**

Query symmetric cryptographic statistics for a specific instance.
查询特定实例的对称加密统计信息。

CpaStatus CPA_DEPRECATED cpaCySymQueryStats( const CpaInstanceHandle *instanceHandle*,
struct **CpaCySymStats** * *pSymStats*

**Deprecated:**
Deprecated:

As of v1.3 of the cryptographic API, this function has been deprecated, replaced by
从加密 API 的 1.3 版开始，此函数已被弃用，由
**cpaCySymQueryStats64()**.
cpaCySymQueryStats64()。

This function will query a specific instance for statistics. The user MUST allocate the CpaCySymStats structure and pass the reference to that into this function call. This function will write the statistic results into the passed in CpaCySymStats structure.
该函数将查询特定实例的统计信息。用户必须分配 CpaCySymStats 结构，并将对该结构的引用传递给这个函数调用。该函数将把统计结果写入传入的 CpaCySymStats 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:**

Yes
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。

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

[out] pSymStats 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, i.e. no asynchronous callback will be generated.
该函数以同步方式运行，即不会生成异步回调。

**See also:**

**另请参见:**

CpaCySymStats
CpaCySymStats

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

Query symmetric cryptographic statistics (64 bit version) for a specific instance.
查询特定实例的对称加密统计信息(64位版本)。

This function will query a specific instance for statistics. The user MUST allocate the CpaCySymStats64 structure and pass the reference to that into this function call. This function will write the statistic results into the passed in CpaCySymStats64 structure.
该函数将查询特定实例的统计信息。用户必须分配 CpaCySymStats64 结构，并将对该结构的引用传递到此函数调用中。该函数将把统计结果写入传入的 CpaCySymStats64 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:**

Yes
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。

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

[out] pSymStats 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, i.e. no asynchronous callback will be generated.
该函数以同步方式运行，即不会生成异步回调。

**See also:**

**另请参见:**

**CpaCySymStats64**
CpaCySymStats64

**CpaStatus** cpaCySymQueryCapabilities ( const **CpaInstanceHandle**
CpaStatus cpaCySymQueryCapabilities ( const CpaInstanceHandle
*instanceHandle*

Returns capabilities of the symmetric API group of a Cryptographic API instance.
返回加密 API 实例的对称 API 组的功能。

This function is used to determine which specific capabilities are supported within the symmetric sub-group of the Cryptographic API.
此函数用于确定加密 API 的对称子组中支持哪些特定功能。

**Context:**
**背景:**

The function shall not be called in an interrupt context.
该函数不应在中断上下文中调用。

**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* Handle to an instance of this API.
[in]此 API 实例的 instanceHandle 句柄。
[out] *pCapInfo* Pointer to capabilities info structure. All fields in the structure are populated by the API instance.
[out]指向功能信息结构的 pCapInfo 指针。结构中的所有字段都由 API 实例填充。

**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_UNSUPPORTED* Function is not supported.

*CPA_STATUS_FAIL 函数失败。传递的 CPA_STATUS_INVALID_PARAM 参数无效。不支持 CPA_STATUS_UNSUPPORTED 函数。*

**Precondition:**

**前提条件:**

The instance has been initialized via the **cpaCyStartInstance** function.
实例已通过初始化 cpaCyStartInstance

**Postcondition:**
**后置条件:**

None
没有人

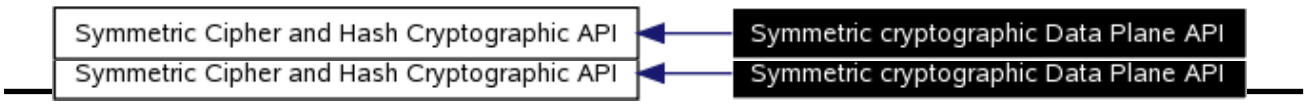
# 14 Symmetric cryptographic Data Plane API

# 15 对称加密数据平面 API

### [Symmetric Cipher and Hash Cryptographic API]

[Symmetric Cipher and Hash Cryptographic API]

Collaboration diagram for Symmetric cryptographic Data Plane API:
对称加密数据平面 API 的协作图:




## 15.1 Detailed Description
## 15.2 详细描述

**File: cpa_cy_sym_dp.h**

文件:cpa_cy_sym_dp.h

These data structures and functions specify the Data Plane API for symmetric cipher, hash, and combined cipher and hash operations.
这些数据结构和函数为对称密码、散列以及组合密码和散列操作指定了数据平面 API。

This API is recommended for data plane applications, in which the cost of offload - that is, the cycles consumed by the driver in sending requests to the hardware, and processing responses - needs to be minimized. In particular, use of this API is recommended if the following constraints are acceptable to your application:
此 API 推荐用于数据层应用，在这种应用中，卸载成本(即驱动程序向硬件发送请求和处理响应所消耗的周期)需要最小化。特别是，如果您的应用程序可以接受以下约束，建议使用此 API:

- Thread safety is not guaranteed. Each software thread should have access to its own unique instance (CpaInstanceHandle) to avoid contention.
- 线程安全性没有保证。每个软件线程都应该能够访问自己唯一的实例(CpaInstanceHandle)以避免争用。
- Polling is used, rather than interrupts (which are expensive). Implementations of this API will provide a function (not defined as part of this API) to read responses from the hardware response queue and dispatch callback functions, as specified on this API.
- 使用轮询，而不是中断(这是昂贵的)。该 API 的实现将提供一个函数(未定义为该 API 的一部分)来从硬件响应队列中读取响应，并根据该 API 的规定分派回调函数。
- Buffers and buffer lists are passed using physical addresses, to avoid virtual to physical address
- 使用物理地址传递缓冲区和缓冲区列表，以避免虚拟地址到物理地址

translation costs.
翻译费用。

- For GCM and CCM modes of AES, when performing decryption and verification, if verification fails, then the message buffer will NOT be zeroed. (This is a consequence of using physical addresses for the buffers.)
- 对于 AES 的 GCM 和 CCM 模式，当执行解密和验证时，如果验证失败，则消息缓冲区不会被清零。(这是为缓冲区使用物理地址的结果。)
- The ability to enqueue one or more requests without submitting them to the hardware allows for
- 将一个或多个请求排队而不提交给硬件的能力允许
  certain costs to be amortized across multiple requests.
  某些成本将在多个请求中分摊。
- Only asynchronous invocation is supported.
- 仅支持异步调用。
- There is no support for partial packets.
- 不支持部分数据包。
- Implementations may provide certain features as optional at build time, such as atomic counters.
- 实现可能在构建时提供某些可选的特性，比如原子计数器。
- The "default" instance (**CPA_INSTANCE_HANDLE_SINGLE**) is not supported on this API. The specific handle should be obtained using the instance discovery functions (**cpaCyGetNumInstances**, **cpaCyGetInstances**).
- “默认”实例(CPA_INSTANCE_HANDLE_SINGLEcpaCyGetNumInstancescpaCyGetInstances

**Note:**

**注意:**

Perf

ormance Trade-Offs Different implementations of this API may have different performance trade-offs; please refer to the documentation for your implementation for details. However, the following concepts informed the definition of this API.
性能权衡这个 API 的不同实现可能有不同的性能权衡；有关详细信息，请参考您的实现文档。但是，以下概念提供了该 API 的定义。

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

API 区分将请求入队和将该请求实际提交给加密加速引擎来执行。这允许多个请求排队(单独或成批)，然后在单个操作中提交所有排队的请求。基本原理是，在一些(尤其是基于硬件的)实现中，提交操作是昂贵的；例如，它可能导致 MMIO 指令。API 允许将这一成本分摊到多个请求中。这种请求的精确数量可以调整以获得最佳性能。

Specifically:
具体来说:

9.1 Detailed Description

9.2 详细描述

- The function **cpaCySymDpEnqueueOp** allows one request to be enqueued, and optionally for that request (and all previously enqueued requests) to be submitted.

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

## 9.3 Data Structures
## 9.4 数据结构

- struct _**CpaCySymDpOpData**

- 结构体_CpaCySymDpOpData

## 9.5 Typedefs
## 9.6 类型定义

- typedef void * **CpaCySymDpSessionCtx**

- typedef void *CpaCySymDpSessionCtx
- typedef _**CpaCySymDpOpData CpaCySymDpOpData**
- 数据类型说明_CpaCySymDpOpData CpaCySymDpOpData
- typedef void(* **CpaCySymDpCbFunc** )(**CpaCySymDpOpData** *pOpData, **CpaStatus** status,
- typedef void(*CpaCySymDpCbFunc CpaCySymDpOpData CpaStatus
  **CpaBoolean** verifyResult)
  CpaBoolean verifyResult)

## 9.7 Functions
## 9.8 功能

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

- CpaStatus cpaCySymDpRegCbFunc （常量 CpaInstanceHandle
  **CpaCySymDpCbFunc** pSymNewCb)
  CpaCySymDpCbFunc pSymNewCb)
- **CpaStatus cpaCySymDpSessionCtxGetSize** (const **CpaInstanceHandle** instanceHandle, const
- CpaStatus cpaCySymDpSessionCtxGetSize （常量 CpaInstanceHandle
  **CpaCySymSessionSetupData** *pSessionSetupData, **Cpa32U** *pSessionCtxSizeInBytes)
  CpaCySymSessionSetupData *pSessionSetupData, Cpa32U
- **CpaStatus cpaCySymDpSessionCtxGetDynamicSize** (const **CpaInstanceHandle** instanceHandle, const **CpaCySymSessionSetupData** *pSessionSetupData, **Cpa32U** *pSessionCtxSizeInBytes)
- CpaStatus cpaCySymDpSessionCtxGetDynamicSize （常量 CpaInstanceHandle
  CpaCySymSessionSetupData Cpa32U
- **CpaStatus cpaCySymDpInitSession** (**CpaInstanceHandle** instanceHandle, const
- CpaStatus cpaCySymDpInitSession （CpaInstanceHandle
  **CpaCySymSessionSetupData** *pSessionSetupData, **CpaCySymDpSessionCtx** sessionCtx)
  CpaCySymSessionSetupData *pSessionSetupData, CpaCySymDpSessionCtx

- **CpaStatus cpaCySymDpRemoveSession** (const **CpaInstanceHandle** instanceHandle,
- CpaStatus cpaCySymDpRemoveSession （常量 CpaInstanceHandle **CpaCySymDpSessionCtx** sessionCtx)
  CpaCySymDpSessionCtx sessionCtx)
- **CpaStatus cpaCySymDpEnqueueOp** (**CpaCySymDpOpData** *pOpData, const **CpaBoolean**
- CpaStatus cpaCySymDpEnqueueOp （CpaCySymDpOpData CpaBoolean performOpNow)
  performOpNow)
- **CpaStatus cpaCySymDpEnqueueOpBatch** (const **Cpa32U** numberRequests,
- CpaStatus cpaCySymDpEnqueueOpBatch （常量 Cpa32U **CpaCySymDpOpData** *pOpData[], const **CpaBoolean** performOpNow)
  CpaCySymDpOpData *pOpData[]，常量 CpaBoolean
- **CpaStatus cpaCySymDpPerformOpNow** (**CpaInstanceHandle** instanceHandle)
- CpaStatus cpaCySymDpPerformOpNow （CpaInstanceHandle

---

# 9.9 Data Structure Documentation

# 9.10 数据结构文档

## 9.5.1 _CpaCySymDpOpData Struct Reference

## 9.5.2 _CpaCySymDpOpData 结构引用

### 9.5.2.1 Detailed Description
### 9.5.2.2 详细描述

Operation Data for cryptographic data plane API.
加密数据平面 API 的操作数据。

This structure contains data relating to a request to perform symmetric cryptographic processing on one or more data buffers.
该结构包含与在一个或多个数据缓冲器上执行对称密码处理的请求相关的数据。

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

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

这个结构指向的物理内存至少需要 8 字节对齐。调用代码不应写入或读取所有保留字

段。

**See also:**
**另请参见:**

**cpaCySymDpEnqueueOp**, **cpaCySymDpEnqueueOpBatch**
cpaCySymDpEnqueueOp, cpaCySymDpEnqueueOpBatch

### 9.5.2.3 Data Fields

### 9.5.2.4 数据字段

- **Cpa64U reserved0**
- Cpa64U reserved0
- **Cpa32U cryptoStartSrcOffsetInBytes**
- Cpa32U cryptoStartSrcOffsetInBytes
- **Cpa32U messageLenToCipherInBytes**
- Cpa32U messageLenToCipherInBytes
- **CpaPhysicalAddr iv**
- CpaPhysicalAddr iv
- **Cpa64U reserved1**
- Cpa64U reserved1
- **Cpa32U hashStartSrcOffsetInBytes**
- Cpa32U hashStartSrcOffsetInBytes
- **Cpa32U messageLenToHashInBytes**
- Cpa32U messageLenToHashInBytes
- **CpaPhysicalAddr additionalAuthData**
- CpaPhysicalAddr additionalAuthData
- **CpaPhysicalAddr digestResult**
- CpaPhysicalAddr digestResult
- **CpaInstanceHandle instanceHandle**
- CpaInstanceHandle instanceHandle
- **CpaCySymDpSessionCtx sessionCtx**
- CpaCySymDpSessionCtx sessionCtx
- **Cpa32U ivLenInBytes**
- Cpa32U ivLenInBytes
- **CpaPhysicalAddr srcBuffer**
- CpaPhysicalAddr srcBuffer
- **Cpa32U srcBufferLen**
- Cpa32U srcBufferLen
- **CpaPhysicalAddr dstBuffer**
- CpaPhysicalAddr dstBuffer
- **Cpa32U dstBufferLen**
- Cpa32U dstBufferLen
- **CpaPhysicalAddr thisPhys**
- CpaPhysicalAddr thisPhys
- **Cpa8U** * **pIv**
- Cpa8U *pIv
- **Cpa8U** * **pAdditionalAuthData**
- Cpa8U *pAdditionalAuthData
- void * **pCallbackTag**
- 无效*pCallbackTag

### 9.5.2.5 Field Documentation
### 9.5.2.6 现场文件

**Cpa64U _CpaCySymDpOpData::reserved0**

Cpa64_CpaCySymDpOpData::reserved

Reserved for internal usage.

保留供内部使用。

### Cpa32U _CpaCySymDpOpData::cryptoStartSrcOffsetInBytes

Cpa32_CpaCySymDpOpData::cryptoStartSrcOffsetInByte

Starting point for cipher processing, specified as number of bytes from start of data in the source buffer. The result of the cipher operation will be written back into the buffer starting at this location in the destination buffer.

密码处理的起点，指定为从源缓冲区中数据开始的字节数。加密操作的结果将从目标缓冲区的这个位置开始写回缓冲区。

### Cpa32U _CpaCySymDpOpData::messageLenToCipherInBytes

The message length, in bytes, of the source buffer on which the cryptographic operation will be computed. This must be a multiple of the block size if a block cipher is being used. This is also the same as the result

Cpa32_CpaCySymDpOpData::messageLenToCipherInByte

length.

长度。

**Note:**

**注意:**

In the case of CCM (**CPA_CY_SYM_HASH_AES_CCM**), this value should not include the length of the padding or the length of the MAC; the driver will compute the actual number of bytes over which the encryption will occur, which will include these values.

在 CCM 的情况下(CPA_CY_SYM_HASH_AES_CCM

For AES-GMAC (**CPA_CY_SYM_HASH_AES_GMAC**), this field should be set to 0.

On some implementations, this length may be limited to a 16-bit value (65535 bytes).

对于 AES-GMAC(CPA_CY_SYM_HASH_AES_GMAC

### CpaPhysicalAddr _CpaCySymDpOpData::iv

CpaPhysicalAdd_CpaCySymDpOpData::i

Initialization Vector or Counter. Specifically, this is the physical address of one of the following:

初始化向量或计数器。具体来说，这是以下地址之一的物理地址:

- For block ciphers in CBC mode, or for Kasumi in F8 mode, or for SNOW3G in UEA2 mode, this is the Initialization Vector (IV) value.
- 对于 CBC 模式下的块密码，或 F8 模式下的小霞，或 UEA2 模式下的 SNOW3G，这是初始化向量(IV)值。
- For ARC4, this is reserved for internal usage.
- 对于 ARC4，这是留给内部使用的。
- For block ciphers in CTR mode, this is the counter.
- 对于 CTR 模式下的分组密码，这是计数器。

- For GCM mode, this is either the IV (if the length is 96 bits) or J0 (for other sizes), where J0 is as defined by NIST SP800-38D. Regardless of the IV length, a full 16 bytes needs to be allocated.
- 对于 GCM 模式，这是 IV(如果长度为 96 位)或 J0(对于其他尺寸)，其中 J0 由 NIST SP800-38D 定义。无论 IV 长度如何，都需要分配整整 16 个字节。
- For CCM mode, the first byte is reserved, and the nonce should be written starting at &pIv[1] (to allow space for the implementation to write in the flags in the first byte). Note that a full 16 bytes should be allocated, even though the ivLenInBytes field will have a value less than this. The macro **CPA_CY_SYM_CCM_SET_NONCE** may be used here.
- 对于 CCM 模式，第一个字节是保留的，nonce 应该从&pIv[1]开始写入(为实现留出空间以写入第一个字节中的标志)。请注意，应该分配完整的 16 个字节，即使 ivLenInBytes 字段的值小于该值。宏 CPA_CY_SYM_CCM_SET_NONCE

### Cpa64U _CpaCySymDpOpData::reserved1

Cpa64_CpaCySymDpOpData::reserved

Reserved for internal usage.

保留供内部使用。

### Cpa32U _CpaCySymDpOpData::hashStartSrcOffsetInBytes

Cpa32_CpaCySymDpOpData::hashStartSrcOffsetInByte

Starting point for hash processing, specified as number of bytes from start of packet in source buffer.

哈希处理的起点，指定为从源缓冲区中的数据包开始算起的字节数。

**Note:**

**注意:**

For CCM and GCM modes of operation, this value in this field is ignored, and the field is reserved for internal usage. The fields **additionalAuthData** and **pAdditionalAuthData** should be set instead.

对于 CCM 和 GCM 工作模式，该域中的值被忽略，该域保留供内部使用。田野 additionalAuthData pAdditionalAuthData

For AES-GMAC (**CPA_CY_SYM_HASH_AES_GMAC**) mode of operation, this field specifies the start of the AAD data in the source buffer.

对于 AES-GMAC(CPA_CY_SYM_HASH_AES_GMAC

### Cpa32U _CpaCySymDpOpData::messageLenToHashInBytes

Cpa32_CpaCySymDpOpData::messageLenToHashInByte

The message length, in bytes, of the source buffer that the hash will be computed on.

将计算哈希的源缓冲区的消息长度(以字节为单位)。

**Note:**

**注意:**

For CCM and GCM modes of operation, this value in this field is ignored, and the field is reserved for internal usage. The fields **additionalAuthData** and **pAdditionalAuthData** should be set instead.

对于 CCM 和 GCM 工作模式，该域中的值被忽略，该域保留供内部使用。田野 additionalAuthData pAdditionalAuthData

For AES-GMAC (**CPA_CY_SYM_HASH_AES_GMAC**) mode of operation, this field specifies the

length of the AAD data in the source buffer.
对于 AES-GMAC(CPA_CY_SYM_HASH_AES_GMAC

On some implementations, this length may be limited to a 16-bit value (65535 bytes).
在一些实施方式中，该长度可能被限制为 16 位值(65535 字节)。

**CpaPhysicalAddr _CpaCySymDpOpData::additionalAuthData**

Physical address of the Additional Authenticated Data (AAD), which is needed for authenticated cipher mechanisms (CCM and GCM), and to the IV for SNOW3G authentication

CpaPhysicalAdd_CpaCySymDpOpData::additionalAuthDat

(**CPA_CY_SYM_HASH_SNOW3G_UIA2**). For other authentication mechanisms, this value is ignored, and （CPA_CY_SYM_HASH_SNOW3G_UIA2

the field is reserved for internal usage.
该字段保留供内部使用。

The length of the data pointed to by this field is set up for the session in the **CpaCySymHashAuthModeSetupData** structure as part of the **cpaCySymDpInitSession** function call. This length must not exceed 240 bytes.
此字段指向的数据长度是在中为进程设置的 CpaCySymHashAuthModeSetupData cpaCySymDpInitSession

If AAD is not used, this address must be set to zero.
如果不使用 AAD，该地址必须设置为零。

Specifically for CCM (**CPA_CY_SYM_HASH_AES_CCM**) and GCM (**CPA_CY_SYM_HASH_AES_GCM**), the caller should be setup as described in the same way as the corresponding field, pAdditionalAuthData, on the "traditional" API (see the **CpaCySymOpData**).
专门针对 CCM(CPA_CY_SYM_HASH_AES_CCMCPA_CY_SYM_HASH_AES_GCMCpaCySymOpData

**Note:**
**注意：**

For AES-GMAC (**CPA_CY_SYM_HASH_AES_GMAC**) mode of operation, this field is not used and should be set to 0. Instead the AAD data should be placed in the source buffer.
对于 AES-GMAC(CPA_CY_SYM_HASH_AES_GMAC

**CpaPhysicalAddr _CpaCySymDpOpData::digestResult**

If the digestIsAppended member of the **CpaCySymSessionSetupData** structure is NOT set then this is the physical address of the location where the digest result should be inserted (in the case of digest generation)

CpaPhysicalAdd_CpaCySymDpOpData::digestResulCpaCySymSessionSetupData
or where the purported digest exists (in the case of digest verification).
或者声称的摘要存在的地方(在摘要验证的情况下)。

At session registration time, the client specified the digest result length with the digestResultLenInBytes member of the **CpaCySymHashSetupData** structure. The client must allocate at least digestResultLenInBytes of physically contiguous memory at this location.
在会话注册时，客户端用 CpaCySymHashSetupData

For digest generation, the digest result will overwrite any data at this location.
对于摘要生成，摘要结果将覆盖此位置的任何数据。

**Note:**
**注意:**

For GCM (**CPA_CY_SYM_HASH_AES_GCM**), for "digest result" read "authentication tag T".
对于 GCM(CPA_CY_SYM_HASH_AES_GCM

If the digestIsAppended member of the **CpaCySymSessionSetupData** structure is set then this value is ignored and the digest result is understood to be in the destination buffer for digest generation, and in the source buffer for digest verification. The location of the digest result in this case is immediately following the region over which the digest is computed.
如果 CpaCySymSessionSetupData

**CpaInstanceHandle _CpaCySymDpOpData::instanceHandle**

CpaInstanceHandl_CpaCySymDpOpData::instanceHandl

Instance to which the request is to be enqueued.
请求要排队到的实例。

**Note:**
**注意:**

A callback function must have been registered on the instance using
必须使用在实例上注册了回调函数
**cpaCySymDpRegCbFunc**.
cpaCySymDpRegCbFunc。

**CpaCySymDpSessionCtx _CpaCySymDpOpData::sessionCtx**

CpaCySymDpSessionCt_CpaCySymDpOpData::sessionCt

Session context specifying the cryptographic parameters for this request.
为该请求指定加密参数的会话上下文。

**Note:**
**注意:**

The session must have been created using
该会话必须是使用

**cpaCySymDpInitSession**.
cpaCySymDpInitSession。

### Cpa32U _CpaCySymDpOpData::ivLenInBytes

Cpa32_CpaCySymDpOpData::ivLenInByte

Length of valid IV data pointed to by the pIv parameter.
pIv 参数指向的有效 IV 数据的长度。

- For block ciphers in CBC mode, or for Kasumi in F8 mode, or for SNOW3G in UEA2 mode, this is the length of the IV (which must be the same as the block length of the cipher).
- 对于 CBC 模式下的分组密码，或 F8 模式下的小霞，或 UEA2 模式下的 SNOW3G，这是 IV 的长度(必须与密码的分组长度相同)。
- For block ciphers in CTR mode, this is the length of the counter (which must be the same as the block length of the cipher).
- 对于 CTR 模式下的分组密码，这是计数器的长度(必须与密码的分组长度相同)。
- For GCM mode, this is either 12 (for 96-bit IVs) or 16, in which case pIv points to J0.
- 对于 GCM 模式，这是 12(对于 96 位 IVs)或 16，在这种情况下 pIv 指向 J0。
- For CCM mode, this is the length of the nonce, which can be in the range 7 to 13 inclusive.
- 对于 CCM 模式，这是随机数的长度，可以在 7 到 13 的范围内，包括 7 和 13。

### CpaPhysicalAddr _CpaCySymDpOpData::srcBuffer

CpaPhysicalAdd_CpaCySymDpOpData::srcBuffe

Physical address of the source buffer on which to operate. This is either:
要操作的源缓冲区的物理地址。这要么是:

- The location of the data, of length srcBufferLen; or,
- 数据的位置，长度为 srcBufferLen 或者，
- If srcBufferLen has the special value **CPA_DP_BUFLIST**, then srcBuffer contains the location where a **CpaPhysBufferList** is stored. In this case, the CpaPhysBufferList MUST be aligned on an 8-byte boundary.
- 如果 srcBufferLen 具有特殊值 CPA_DP_BUFLISTCpaPhysBufferList
- For optimum performance, the buffer should only contain the data region that the cryptographic
- 为了获得最佳性能，缓冲区应该只包含加密
  operation(s) must be performed on. Any additional data in the source buffer may be copied to the destination buffer and this copy may degrade performance.
  操作必须在上执行。源缓冲区中的任何附加数据都可能被复制到目标缓冲区，这种复制可能会降低性能。