Arm® CryptoCell-703

Revision: r0p0

Software Developers Manual

Issue 01

101731

Non-Confidential

Arm® CryptoCell-703

Software Developers Manual

Release information

Document history

Issue	Date	Confidentiality	Change
0000-01	17-July-2019	Non-Confidential	First official release for r0p0

Non-Confidential Proprietary Notice

This document is protected by copyright and other related rights and the practice or implementation of the information contained in this document may be protected by one or more patents or pending patent applications. No part of this document may be reproduced in any form by any means without the express prior written permission of Arm. **No license, express or implied, by estoppel or otherwise to any intellectual property rights is granted by this document unless specifically stated.**

Your access to the information in this document is conditional upon your acceptance that you will not use or permit others to use the information for the purposes of determining whether implementations infringe any third party patents.

THIS DOCUMENT IS PROVIDED “AS IS”. ARM PROVIDES NO REPRESENTATIONS AND NO WARRANTIES, EXPRESS, IMPLIED OR STATUTORY, INCLUDING, WITHOUT LIMITATION, THE IMPLIED WARRANTIES OF MERCHANTABILITY, SATISFACTORY QUALITY, NON-INFRINGEMENT OR FITNESS FOR A PARTICULAR PURPOSE WITH RESPECT TO THE DOCUMENT. For the avoidance of doubt, Arm makes no representation with respect to, and has undertaken no analysis to identify or understand the scope and content of, patents, copyrights, trade secrets, or other rights.

This document may include technical inaccuracies or typographical errors.

TO THE EXTENT NOT PROHIBITED BY LAW, IN NO EVENT WILL ARM BE LIABLE FOR ANY DAMAGES, INCLUDING WITHOUT LIMITATION ANY DIRECT, INDIRECT, SPECIAL, INCIDENTAL, PUNITIVE, OR CONSEQUENTIAL DAMAGES, HOWEVER CAUSED AND REGARDLESS OF THE THEORY OF LIABILITY, ARISING OUT OF ANY USE OF THIS DOCUMENT, EVEN IF ARM HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

This document consists solely of commercial items. You shall be responsible for ensuring that any use, duplication or disclosure of this document complies fully with any relevant export laws and regulations to assure that this document or any portion thereof is not exported, directly or indirectly, in violation of such export laws. Use of the word “partner” in reference to Arm's customers is not intended to create or refer to any partnership relationship with any other company. Arm may make changes to this document at any time and without notice.

If any of the provisions contained in these terms conflict with any of the provisions of any click through or signed written agreement covering this document with Arm, then the click through or signed written agreement prevails over and supersedes the conflicting provisions of these terms. This document may be translated into other languages for convenience, and you agree that if there is any conflict between the English version of this document and any translation, the terms of the English version of the Agreement shall prevail.

Arm Limited. Company 02557590 registered in England.

110 Fulbourn Road, Cambridge, England CB1 9NJ.

LES-PRE-20349

Confidentiality Status

This document is Non-Confidential. The right to use, copy and disclose this document may be subject to license restrictions in accordance with the terms of the agreement entered into by Arm and the party that Arm delivered this document to.

Unrestricted Access is an Arm internal classification.

Product Status

The information in this document is Final, that is for a developed product.

Web Address

http://www.arm.com

Contents

1 Introduction

1.1 Product revision status

The r*m*p*n* identifier indicates the revision status of the product described in this book, for example, r*1*p*2*, where:

r*m* Identifies the major revision of the product, for example, r*1*.

p*n* Identifies the minor revision or modification status of the product, for example, p*2*.

1.2 Intended audience

This document is written for programmers using the CryptoCell-703 cryptographic APIs.

Familiarity with the basics of security and cryptography is assumed.

1.3 Conventions

The following subsections describe conventions used in Arm documents.

1.3.1 Glossary

The Arm Glossary is a list of terms used in Arm documentation, together with definitions for those terms. The Arm Glossary does not contain terms that are industry standard unless the Arm meaning differs from the generally accepted meaning.

See the **Arm® Glossary** for more information.

1.3.2 Typographical conventions

Convention	Use
italic	Introduces special terminology, denotes cross-references, and citations.
bold	Highlights interface elements, such as menu names. Denotes signal names. Also used for terms in descriptive lists, where appropriate.
`monospace`	Denotes text that you can enter at the keyboard, such as commands, file and program names, and source code.

Convention	Use
`Monospace` **`bold`**	Denotes language keywords when used outside example code.
`monospace italic`	Denotes arguments to monospace text where the argument is to be replaced by a specific value.
`monospace underline`	Denotes a permitted abbreviation for a command or option. You can enter the underlined text instead of the full command or option name.
<and>	Encloses replaceable terms for assembler syntax where they appear in code or code fragments. For example: `MRC p15, 0, <Rd>, <CRn>, <CRm>, <Opcode_2>`
SMALL CAPITALS	Used in body text for a few terms that have specific technical meanings, that are defined in the Arm® Glossary. For example, IMPLEMENTATION DEFINED, IMPLEMENTATION SPECIFIC, UNKNOWN, and UNPREDICTABLE.
⚠	Caution
✋	Warning
✎	Note

1.4 Additional reading

This document contains information that is specific to this product. See the following documents for other relevant information:

Table 1-1 Arm publications

Document name	Document ID	Licensee only Y/N
Arm® CryptoCell-703 Software Integrators Manual	101730	Y
Arm® CryptoCell-703 Software Release Notes	PJDOC-1779577084-15630	Y
Arm® TRNG Characterization Application Note	100685	Y
Arm® AMBA® AXI and ACE Protocol Specification, February 2013	ARM IHI 0022F	N

Document name	Document ID	Licensee only Y/N
Arm® Trusted Base System Architecture V1: System Software on Arm	ARM DEN 0007B-2	N
Power State Coordination Interface Platform Design Document	ARM DEN 0022D	N
Arm® Platform Security Architecture Trusted Base System Architecture for ARMv8-M	ARM DEN 0062A-B1	N

Table 1-2 Other publications

Document ID	Document name
ANSI X3.92-1981	Data Encryption Algorithm
ANSI X3.106-1983	Data Encryption Algorithm – Modes of Operation
ANSI X9.31-1988	Public Key Cryptography Using Reversible Algorithms for the Financial Services Industry (rDSA)
ANSI X9.42-2003	Public Key Cryptography for the Financial Services Industry: Agreement of Symmetric Keys Using Discrete Logarithm Cryptography
ANSI X9.52-1998	Triple Data Encryption Algorithm Modes of Operation
BSI AIS-31	Functionality Classes and Evaluation Methodology for True Random Number Generators, version 3.1, September 2001
_	ChinaDRM Compliance Rules and Robustness Rules, December 2016
_	ChinaDRM lab: A description of ChinaDRM implementation (2016)
FIPS Publication 46-3	Data Encryption Standard (DES)
FIPS Publication 81	DES Modes of Operation
FIPS Publication 140IG	Implementation Guidance for FIPS PUB 140-2 and the Cryptographic Module Validation Program (November 2015
FIPS Publication 140-2	Security Requirements for Cryptographic Modules
FIPS Publication 180-4	Secure Hash Standard (SHS)
FIPS Publication 186-4	Digital Signature Standard (DSS)
FIPS Publication 197	Advanced Encryption Standard
FIPS Publication 198-1:	The Keyed-Hash Message Authentication Code (HMAC)
GM/T 0005-2012	Chinese randomness test specification
GM/T 0009-2012 SM2	Cryptography algorithm application specification Chinese academy of science
GM/T 0009-2012 SM2	Cryptographic algorithm encryption signature message syntax specification Chinese academy of science
ISO/IEC 9797-1	Message Authentication Codes (MACs) -- Part 1: Mechanisms using a block ciphe

Document ID	Document name
ISO/IEC 18033-2:2006	Information technology -- Security techniques -- Encryption algorithms -- Part 2: Asymmetric cipher
IEEE 1363-2000	IEEE Standard for Standard Specifications for Public-Key Cryptography
NIST SP 800-22	A Statistical Test Suite for Random and Pseudorandom Number Generators for Cryptographic Application
NIST SP 800-38A	Recommendation for Block Cipher Modes of Operation: Methods and Techniques
NIST SP 800-38B	Recommendation for Block Cipher Modes of Operation: the CMAC Mode for Authentication
NIST SP 800-38C	Recommendation for Block Cipher Modes of Operation: the CCM Mode for Authentication and Confidentialit
NIST SP 800-38D	Recommendation for Block Cipher Modes of Operation: Galois/Counter Mode (GCM) and GMAC
NIST SP 800-38E	Recommendation for Block Cipher Modes of Operation: the XTSAES Mode for Confidentiality on Storage Devices
NIST SP 800-56A Rev. 2	Recommendation for Pair-Wise Key Establishment Schemes Using Discrete Logarithm Cryptography
NIST SP 800-38F	Recommendation for Block Cipher Modes of Operation: Methods for Key Wrapping
NIST SP 800-57A Rev. 4	Recommendation for Key Management – Part 1: General
NIST SP 800-90A	Recommendation for Random Number Generation Using Deterministic Random Bit Generators – App C
NIST SP 800-90B	Recommendation for the Entropy Sources Used for Random Bit Generation, January 2018
NIST SP 800-90C	Recommendation for Random Bit Generator (RBG) Constructions
NIST SP 800-108	Recommendation for Key Derivation Using Pseudorandom Functions
NIST SP 800-135 Rev. 1	Recommendation for Existing Application-Specific Key Derivation Functions
PKCS #1 v1.5	RSA Encryption
PKCS #1 v2.1	RSA Cryptography Specifications
PKCS #3	Diffie Hellman Key Agreement Standard
PKCS #7 v1	Cryptographic Message Syntax Standard
RFC 2104	HMAC: Keyed-Hashing for Message Authentication
RFC 3394	Advanced Encryption Standard (AES) Key Wrap Algorithm
RFC 3566	The AES-XCBC-MAC-96 Algorithm and Its Use with IPsec
RFC 3566	Using Advanced Encryption Standard (AES) Counter Mode With IPsec Encapsulating Security Payload (ESP

Document ID	Document name
RFC 4106	The Use of Galois/Counter Mode (GCM) in IPsec Encapsulating Security Payload (ESP)
RFC 4309	Using Advanced Encryption Standard (AES) CCM Mode with IPsec Encapsulating Security Payload (ESP)
RFC 4543	The Use of Galois Message Authentication Code (GMAC) in IPsec ESP and AH
RFC 5280	Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profil
RFC 5652	Cryptograph.ic Message syntax, section 6.3 (September 2009
RFC 5869	HMAC-based Extract-and-Expand Key Derivation Function (HKDF)
S**ECG** SEC 2 v1:	Recommended Elliptic Curve Domain Parameters
SECG SEC2 v2	Recommended Elliptic Curve Domain Parameters
SECG SEC1	Standards for Efficient Cryptography Group (SECG): SEC1 Elliptic Curve Cryptography
JESD223C	Universal Flash Storage Host Controller Interface (UFSHCI), Version 2.1
JESD223C	X9.62-2005: Public Key Cryptography for the Financial Services Industry, The Elliptic Curve Digital Signature Algorithm (ECDSA)
X9.63-2011	Public Key Cryptography for the Financial Services Industry – Key Agreement and Key Transport Using Elliptic Curve Cryptography

1.5 Feedback

Arm welcomes feedback on this product and its documentation.

1.5.1 Feedback on this product

If you have any comments or suggestions about this product, contact your supplier and give:

- The product name.
- The product revision or version.
- An explanation with as much information as you can provide. Include symptoms and diagnostic procedures if appropriate.

1.5.2 Feedback on content

If you have comments on content, send an e-mail to **errata@arm.com** and give:

- The title Arm® CryptoCell-703 Software Developers Manual.
- The number 101731.
- If applicable, the page number(s) to which your comments refer.
- A concise explanation of your comments.

Arm also welcomes general suggestions for additions and improvements.

Arm tests the PDF only in Adobe Acrobat and Acrobat Reader, and cannot guarantee the quality of the represented document when used with any other PDF reader.

2 Runtime APIs

2.1 CryptoCell -703 runtime software API overview

This documentation describes the runtime APIs provided by Arm CryptoCell-703. It provides you with all the information necessary for integrating and using the runtime APIs in the target environment. This documentation also contains the integration tests that you will need to run. The API layer enables use of the following algorithms and features:

- Public Key Cryptographic Algorithm SM2 Based on Elliptic Curves
- SM3 Cryptographic Hash Algorithm
- SM4 Cryptographic Block Cipher
- True Random Number generator
- Content Protection Policy keys
- Power management

This documentation is automatically generated from the source code using Doxygen.

For more information on Doxygen, see `http://www.doxygen.nl/manual/index.html.`

The **Modules** section introduces the high-level module concepts used throughout this documentation.

2.2 Modules

Here is a list of all modules:

- Chinese certification cryptographic APIs
    - Chinese certification cryptographic definitions
    - Chinese certification errors
- CryptoCell ECC APIs
    - CryptoCell APIs for generation of ECC private and public keys
    - CryptoCell ECC specific errors
    - CryptoCell ECPKI type definitions
- CryptoCell PAL APIs
    - CERT definitions
    - CryptoCell PAL DMA related APIs
    - CryptoCell PAL TRNG APIs
    - CryptoCell PAL abort operations
    - CryptoCell PAL entry or exit point APIs
    - CryptoCell PAL logging APIs and definitions
    - CryptoCell PAL memory Barrier APIs
    - CryptoCell PAL memory mapping APIs
    - CryptoCell PAL memory operations
    - CryptoCell PAL mutex APIs
    - CryptoCell PAL platform-dependent compiler-specific definitions
    - CryptoCell PAL power-management APIs
    - CryptoCell platform-dependent PAL layer definitions and types
    - Specific errors of the CryptoCell PAL APIs
- CryptoCell definitions
    - CryptoCell AES type definitions
    - CryptoCell general certification definitions
    - CryptoCell hash type definitions
    - CryptoCell library enums and definitions
    - CryptoCell register APIs
    - General base error codes for CryptoCell
    - PKA enums and definitions
    - Specific errors of the CryptoCell utility module APIs
    - bit-field operations macros

- SM2 APIs
- SM3 APIs
    - CryptoCell SM3 specific errors
    - CryptoCell SM3 type definitions
- SM4 APIs
    - CryptoCell SM4 specific errors
    - CryptoCell SM4 type definitions
- TRNG APIs
    - CryptoCell TRNG specific errors
    - Random number definitions
        - CryptoCell random-number generation definitions.
        - CryptoCell random-number specific errors
        - CryptoCell true-random-number generation definitions.
    - TRNG API definition

2.3 Data structures

The following are the data structures that are part of the delivery:

- **CC_PalTrngParams_t**
- **CC_Sm2KeContext_t**
- **CCAesHwKeyData_t**
- **CCAesUserContext_t**
- **CCAesUserKeyData_t**
- **CCAxiAceConst_t**
- **CCAxiFields_t**
- **CCAximCacheParams_t**
- **CCCertKatContext_t**
- **CCEcdhFipsKatContext_t**
- **CCEcdhTempData_t**
- **CCEcdsaFipsKatContext_t**
- **CCEcdsaSignUserContext_t**
- **CCEcdsaVerifyUserContext_t**
- **CCEciesTempData_t**
- **CCEcpkiBuildTempData_t**

- CCEcpkiDomain_t
- CCEcpkiKgFipsContext_t
- CCEcpkiKgTempData_t
- CCEcpkiPointAffine_t
- CCEcpkiPrivKey_t
- CCEcpkiPublKey_t
- CCEcpkiUserPrivKey_t
- CCEcpkiUserPublKey_t
- CCHashUserContext_t
- CCPalDmaBlockInfo_t
- CCRndContext_t
- CCRndState_t
- CCSm2FipsKatContext_t
- CCSm2KeyGenCHCertContext_t
- CCSm3UserContext_t
- CCSm4UserContext_t
- CCTrngParams_t
- CCTrngState_t
- CCTrngWorkBuff_t
- EcdsaSignContext_t
- EcdsaVerifyContext_t

2.4 File list

The following table lists the files that are part of the delivery, and their descriptions:

Table 2-1 List of files

Filename	Description
`cc_aes_defs.h`	This file contains the type definitions that are used by the CryptoCell AES APIs.
`cc_aes_defs_proj.h`	This file contains definitions that are used in the CryptoCell AES APIs.
`cc_axi_ctrl.h`	This file contains the AXI configuration control definitions.
`cc_bitops.h`	This file defines bit-field operations macros.
`cc_cert_ctx.h`	This file contains definitions that are required for CryptoCell's certification (FIPS or Chinese).
`cc_chinese_cert.h`	This file contains definitions and APIs that are used in the CryptoCell Chinese Certification module.

Filename	Description
cc_chinese_cert_error.h	This file contains error codes definitions for CryptoCell Chinese certification module.
cc_ecpki_build.h	This file defines functions for building key structures used in Elliptic Curves Cryptography (ECC).
cc_ecpki_domain_sm2.h	This file defines the SM2 get domain API.
cc_ecpki_error.h	This file contains the definitions of the CryptoCell ECPKI errors.
cc_ecpki_kg.h	This file defines the API for generation of ECC private and public keys.
cc_ecpki_types.h	This file contains all the type definitions that are used for the CryptoCell ECPKI APIs.
cc_ecpki_types_common.h	This file contains all the type definitions that are used for the CryptoCell ECPKI APIs.
cc_error.h	This file defines the error return code types and the numbering spaces for each module of the layers listed.
cc_hash_defs.h	This file contains definitions of the CryptoCell hash APIs.
cc_hash_defs_proj.h	This file contains HASH definitions.
cc_lib.h	This file contains all the enums and definitions that are used for the CryptoCell library initiation and finish APIs, as well as the APIs themselves.
cc_pal_abort.h	This file includes all PAL APIs.
cc_pal_barrier.h	This file contains the definitions and APIs for memory-barrier implementation.
cc_pal_cert.h	This file contains definitions that are used by the CERT related APIs. The implementation of these functions need to be replaced according to the Platform and TEE_OS.
cc_pal_compiler.h	This file contains CryptoCell PAL platform-dependent compiler-related definitions.
cc_pal_dma.h	This file contains definitions that are used for DMA-related APIs. The implementation of these functions need to be replaced according to the platform and OS.
cc_pal_dma_defs.h	This file contains the platform-dependent DMA definitions.
cc_pal_error.h	This file contains the error definitions of the platform-dependent PAL APIs.
cc_pal_init.h	This file contains the PAL layer entry point. It includes the definitions and APIs for PAL initialization and termination.
cc_pal_log.h	This file contains the PAL layer log definitions. The log is disabled by default.
cc_pal_mem.h	This file contains functions for memory operations.
cc_pal_memmap.h	This file contains functions for memory mapping.
cc_pal_mutex.h	This file contains functions for resource management (mutex operations).

Filename	Description
`cc_pal_pm.h`	This file contains the definitions and APIs for power-management implementation.
`cc_pal_trng.h`	This file contains APIs for retrieving TRNG user parameters.
`cc_pal_types.h`	This file contains platform-dependent definitions and types of the PAL layer.
`cc_pal_types_plat.h`	This file contains basic platform-dependent type definitions.
`cc_pka_defs_hw.h`	This file contains all the enums and definitions that are used in the PKA related code.
`cc_pka_hw_plat_defs.h`	Contains the enums and definitions that are used in the PKA code.
`cc_regs.h`	This file contains macro definitions for accessing Arm CryptoCell registers.
`cc_rnd_common.h`	This file contains the CryptoCell random-number generation APIs.
`cc_rnd_common_trng.h`	This file contains the CryptoCell true-random-number generation definitions. The true-random-number generation module defines the database used for the TRNG operations.
`cc_rnd_error.h`	This file contains the definitions of the CryptoCell RND errors.
`cc_sm2.h`	This file defines the APIs that support the SM2 functions.
`cc_sm3.h`	This file contains all the enums and definitions that are used for the CryptoCell SM3 APIs, as well as the APIs themselves.
`cc_sm3_defs.h`	This file contains definitions of the CryptoCell SM3 APIs.
`cc_sm3_defs_proj.h`	This file contains SM3 definitions.
`cc_sm3_error.h`	This file contains the definitions of the CryptoCell SM3 errors.
`cc_sm4.h`	This file contains all the enums and definitions that are used for the CryptoCell SM4 APIs, as well as the APIs themselves.
`cc_sm4_defs.h`	This file contains the type definitions that are used by the CryptoCell SM4 APIs.
`cc_sm4_defs_proj.h`	This file contains definitions that are used in the CryptoCell SM4 APIs.
`cc_sm4_error.h`	This file contains the definitions of the CryptoCell SM4 errors.
`cc_trng_error.h`	This file contains the definitions of the CryptoCell TRNG errors.
`cc_trng_fe.h`	This file contains API and definitions for generating TRNG buffer in full entropy mode.
`cc_util_error.h`	This file contains the error definitions of the CryptoCell utility APIs.

2.5 Module Documentation

2.5.1 CERT definitions

Contains definitions that are used by the CERT related APIs. The implementation of these functions need to be replaced according to the Platform and TEE_OS.

2.5.1.1 Functions

- **CCError_t CC_PalCertGetState** (uint32_t *pCertState)

  This function purpose is to get the CERT state.

- **CCError_t CC_PalCertGetError** (uint32_t *pCertError)

  This function purpose is to get the CERT error.

- **CCError_t CC_PalCertGetTrace** (uint32_t *pCertTrace)

  This function purpose is to get the CERT trace.

- **CCError_t CC_PalCertSetState** (uint32_t certState)

  This function purpose is to set the CERT state.

- **CCError_t CC_PalCertSetError** (uint32_t certError)

  This function purpose is to set the CERT error.

- **CCError_t CC_PalCertSetTrace** (uint32_t certTrace)

  This function purpose is to set the CERT trace.

- **CCError_t CC_PalCertWaitForReeStatus** (void)

  This function purpose is to wait for CERT interrupt. After GPR0 (==CERT) interrupt is detected, clear the interrupt in ICR, and call CC_FipsIrqHandle.

- **CCError_t CC_PalCertStopWaitingRee** (void)

  This function purpose is to stop waiting for REE CERT interrupt. since TEE lib is terminating.

2.5.1.2 Function documentation

2.5.1.2.1 CCError_t **CC_PalCertGetError (uint32_t * *pCertError*)**

Returns:

Zero on success.

A non-zero value on failure.

2.5.1.2.2 CCError_t **CC_PalCertGetState (uint32_t * *pCertState*)**

Returns:

Zero on success.

A non-zero value on failure.

Parameters:

I/O	Parameter	Description
in	`pCertState`	The address of the buffer to map.

2.5.1.2.3 CCError_t **CC_PalCertGetTrace (uint32_t * *pCertTrace*)**

Returns:

Zero on success.

A non-zero value on failure.

2.5.1.2.4 CCError_t **CC_PalCertSetError (uint32_t *certError*)**

Returns:

Zero on success.

A non-zero value on failure.

2.5.1.2.5 CCError_t **CC_PalCertSetState (uint32_t *certState*)**

Returns:

Zero on success.

A non-zero value on failure.

2.5.1.2.6 CCError_t **CC_PalCertSetTrace (uint32_t *certTrace*)**

Returns:

Zero on success.

A non-zero value on failure.

2.5.1.2.7 CCError_t **CC_PalCertStopWaitingRee (void)**

Returns:

Zero on success.

A non-zero value on failure.

2.5.1.2.8 CCError_t **CC_PalCertWaitForReeStatus (void)**

Returns:

Zero on success.

A non-zero value on failure.

2.5.2 Chinese certification cryptographic APIs

Contains Chinese certification cryptographic APIs and definitions.

2.5.2.1 Modules

- **Chinese certification cryptographic definitions**

  Contains definitions and APIs that are used in the CryptoCell Chinese Certification module.

- **Chinese certification errors**

Contains Chinese certification error definitions.

2.5.3 Chinese certification cryptographic definitions

Contains definitions and APIs that are used in the CryptoCell Chinese Certification module.

2.5.3.1 Macros

- #define **CC_CH_CERT_STATE_NOT_SUPPORTED** 0x0
- #define **CC_CH_CERT_STATE_ERROR** 0x1
- #define **CC_CH_CERT_STATE_SUPPORTED** 0x2
- #define **CC_CH_CERT_STATE_CRYPTO_APPROVED** 0x4
- #define **CC_CH_CERT_CRYPTO_USAGE_SET_APPROVED**() **CC_ChCertCryptoUsageStateSet**(**CC_TEE_CH_CERT_CRYPTO_USAGE_STATE_APPROVED**)
- #define **CC_CH_CERT_CRYPTO_USAGE_SET_NON_APPROVED**() **CC_ChCertCryptoUsageStateSet**(**CC_TEE_CH_CERT_CRYPTO_USAGE_STATE_NON_APPROVED**)

2.5.3.2 typedefs

- typedef uint32_t **CCChCertState_t**

2.5.3.3 Enumerations

- enum **CCChCertError_t** { **CC_TEE_CH_CERT_ERROR_OK** = 0, **CC_TEE_CH_CERT_ERROR_GENERAL**, **CC_TEE_CH_CERT_ERROR_SM4_ECB_PUT**, **CC_TEE_CH_CERT_ERROR_SM4_CBC_PUT**, **CC_TEE_CH_CERT_ERROR_SM4_CTR_PUT**, **CC_TEE_CH_CERT_ERROR_SM3_PUT**, **CC_TEE_CH_CERT_ERROR_SM2_SIGN_PUT**, **CC_TEE_CH_CERT_ERROR_SM2_KEY_GEN_COND**, **CC_TEE_CH_CERT_ERROR_RESERVE32B** = INT32_MAX }
- enum **CCChCertCryptoUsageState_t** { **CC_TEE_CH_CERT_CRYPTO_USAGE_STATE_NON_APPROVED** = 0,

CC_TEE_CH_CERT_CRYPTO_USAGE_STATE_APPROVED,
CC_TEE_CH_CERT_CRYPTO_USAGE_STATE_RESERVE32B = INT32_MAX }

2.5.3.4 Functions

- **CCError_t CC_ChCertErrorGet** (**CCChCertError_t** *pChCertError)

  This function is used to get the current Chinese certification error of the Arm CryptoCell TEE library.

- **CCError_t CC_ChCertStateGet** (**CCChCertState_t** *pChCertState)

  This function is used to get the current state of the Chinese certification state (Chinese certification state set to ON or OFF) and zeroization state of the Arm CryptoCell TEE library.

- **CCError_t CC_ChCertCryptoUsageStateSet** (**CCChCertCryptoUsageState_t** state)

  This function is used to set the permission (approved/non-approved) of the crypto operations in the suspended state of the Arm CryptoCell TEE library.

2.5.3.5 Macro definition documentation

2.5.3.5.1 #define CC_CH_CERT_CRYPTO_USAGE_SET_APPROVED() CC_ChCertCryptoUsageStateSet(CC_TEE_CH_CERT_CRYPTO_USAGE_STATE_APPROVED)

A macro to set the Chinese certification state to approved.

2.5.3.5.2 #define CC_CH_CERT_CRYPTO_USAGE_SET_NON_APPROVED() CC_ChCertCryptoUsageStateSet(CC_TEE_CH_CERT_CRYPTO_USAGE_STATE_NON_APPROVED)

A macro to set the Chinese certification state to not approved.

2.5.3.5.3 #define CC_CH_CERT_STATE_CRYPTO_APPROVED 0x4

State definition of Chinese certification - approved.

2.5.3.5.4 #define CC_CH_CERT_STATE_ERROR 0x1

State definition of Chinese certification - error.

2.5.3.5.5 #define CC_CH_CERT_STATE_NOT_SUPPORTED 0x0

State definition of Chinese certification - unsupported.

2.5.3.5.6 #define CC_CH_CERT_STATE_SUPPORTED 0x2

State definition of Chinese certification - supported.

2.5.3.6 typedef documentation

2.5.3.6.1 typedef uint32_t CCChCertState_t

Definition of Chinese certification state.

2.5.3.7 Enumeration type documentation

2.5.3.7.1 enum CCChCertCryptoUsageState_t

Enumerator:

Enum	Description
CC_TEE_CH_CERT_CRYPTO_USAGE_STATE_NON_APPROVED	Identifies the system as failed the Chinese certifications tests.
CC_TEE_CH_CERT_CRYPTO_USAGE_STATE_APPROVED	Identifies the system as passed the Chinese certifications tests.
CC_TEE_CH_CERT_CRYPTO_USAGE_STATE_RESERVE32B	Reserved error code.

2.5.3.7.2 enum CCChCertError_t

Enumerator:

Enum	Description
CC_TEE_CH_CERT_ERROR_OK	A success indication.
CC_TEE_CH_CERT_ERROR_GENERAL	A general error.
CC_TEE_CH_CERT_ERROR_SM4_ECB_PUT	SM4 ECB tests failure.
CC_TEE_CH_CERT_ERROR_SM4_CBC_PUT	SM4 CBC tests failure.
CC_TEE_CH_CERT_ERROR_SM4_CTR_PUT	SM4 CTR tests failure.
CC_TEE_CH_CERT_ERROR_SM3_PUT	SM3 tests failure.
CC_TEE_CH_CERT_ERROR_SM2_SIGN_PUT	SM2 Sign/Verify tests failure.
CC_TEE_CH_CERT_ERROR_SM2_KEY_GEN_COND	SM2 conditional tests failure.
CC_TEE_CH_CERT_ERROR_RESERVE32B	Reserved error code.

2.5.3.8 Function documentation

2.5.3.8.1 CCError_t **CC_ChCertCryptoUsageStateSet** (CCChCertCryptoUsageState_t ***state***)

Returns:

`CC_OK` on success,

A non-zero value from **cc_chinese_cert_error.h** on failure.

Parameters:

I/O	Parameter	Description
in	state	The state of the cryptographic operations.

2.5.3.8.2 CCError_t **CC_ChCertErrorGet (**CCChCertError_t ***** ***pChCertError*****)**

Returns:

CC_OK on success,

A non-zero value from **cc_chinese_cert_error.h** on failure.

Parameters:

I/O	Parameter	Description
out	pChCertError	The current Chinese certification error of the library.

2.5.3.8.3 CCError_t **CC_ChCertStateGet (**CCChCertState_t ***** ***pChCertState*****)**

Returns:

CC_OK on success,

A non-zero value from **cc_chinese_cert_error.h** on failure.

Parameters:

I/O	Parameter	Description
out	pChCertState	The Chinese certification State of the library (in accordance with the certification state definitions.)

2.5.4 Chinese certification errors

Contains Chinese certification error definitions.

2.5.4.1 Macros

- #define **CC_CH_CERT_ERROR** (**CC_CH_CERT_MODULE_ERROR_BASE** + 0x00UL)

2.5.4.2 Macro definition documentation

2.5.4.2.1 #define CC_CH_CERT_ERROR (CC_CH_CERT_MODULE_ERROR_BASE + 0x00UL)

Chinese Certification module error base address - 0x00F01800

2.5.5 CryptoCell AES type definitions

Contains CryptoCell AES type definitions.

2.5.5.1 Data structures

- struct CCAesUserContext_t
- struct CCAesUserKeyData_t
- struct CCAesHwKeyData_t

2.5.5.2 Macros

- #define CC_AES_USER_CTX_SIZE_IN_WORDS 131
- #define CC_AES_KEY_MAX_SIZE_IN_WORDS 16
- #define CC_AES_KEY_MAX_SIZE_IN_BYTES (CC_AES_KEY_MAX_SIZE_IN_WORDS *sizeof(uint32_t))
- #define CC_AES_CRYPTO_BLOCK_SIZE_IN_WORDS 4
- #define CC_AES_BLOCK_SIZE_IN_BYTES (CC_AES_CRYPTO_BLOCK_SIZE_IN_WORDS *sizeof(uint32_t))
- #define CC_AES_IV_SIZE_IN_WORDS CC_AES_CRYPTO_BLOCK_SIZE_IN_WORDS
- #define CC_AES_IV_SIZE_IN_BYTES (CC_AES_IV_SIZE_IN_WORDS *sizeof(uint32_t))

2.5.5.3 typedefs

- typedef uint8_t CCAesIv_t[CC_AES_IV_SIZE_IN_BYTES]
- typedef uint8_t CCAesKeyBuffer_t[CC_AES_KEY_MAX_SIZE_IN_BYTES]
- typedef struct CCAesUserContext_t CCAesUserContext_t
- typedef struct CCAesUserKeyData_t CCAesUserKeyData_t
- typedef struct CCAesHwKeyData_t CCAesHwKeyData_t

2.5.5.4 Enumerations

- enum CCAesEncryptMode_t { CC_AES_ENCRYPT = 0, CC_AES_DECRYPT = 1, CC_AES_NUM_OF_ENCRYPT_MODES, CC_AES_ENCRYPT_MODE_LAST = 0x7FFFFFFF }
- enum CCAesOperationMode_t { CC_AES_MODE_ECB = 0, CC_AES_MODE_CBC = 1, CC_AES_MODE_CBC_MAC = 2, CC_AES_MODE_CTR = 3, CC_AES_MODE_XCBC_MAC = 4, CC_AES_MODE_CMAC = 5, CC_AES_MODE_XTS = 6, CC_AES_MODE_CBC_CTS = 7, CC_AES_MODE_OFB = 8, CC_AES_MODE_CFB = 9, CC_AES_NUM_OF_OPERATION_MODES, CC_AES_OPERATION_MODE_LAST = 0x7FFFFFFF }
- enum CCAesPaddingType_t { CC_AES_PADDING_NONE = 0, CC_AES_PADDING_PKCS7 = 1, CC_AES_NUM_OF_PADDING_TYPES, CC_AES_PADDING_TYPE_LAST = 0x7FFFFFFF }
- enum CCAesKeyType_t { CC_AES_USER_KEY = 0, CC_AES_PLATFORM_KEY = 1, CC_AES_CUSTOMER_KEY = 2, CC_AES_NUM_OF_KEY_TYPES, CC_AES_KEY_TYPE_LAST = 0x7FFFFFFF }

2.5.5.5 Macro definition documentation

2.5.5.5.1 #define CC_AES_BLOCK_SIZE_IN_BYTES (CC_AES_CRYPTO_BLOCK_SIZE_IN_WORDS *sizeof(uint32_t))

The size of the AES block in bytes.

2.5.5.5.2 #define CC_AES_CRYPTO_BLOCK_SIZE_IN_WORDS 4

The size of the AES block in words.

2.5.5.5.3 #define CC_AES_IV_SIZE_IN_BYTES (CC_AES_IV_SIZE_IN_WORDS *sizeof(uint32_t))

The size of the IV buffer in bytes.

2.5.5.5.4 #define CC_AES_IV_SIZE_IN_WORDS CC_AES_CRYPTO_BLOCK_SIZE_IN_WORDS

The size of the IV buffer in words.

2.5.5.5.5 #define CC_AES_KEY_MAX_SIZE_IN_BYTES (CC_AES_KEY_MAX_SIZE_IN_WORDS *sizeof(uint32_t))

The maximum size of the AES key in bytes.

2.5.5.5.6 #define CC_AES_KEY_MAX_SIZE_IN_WORDS 16

The maximum size of the AES key in words.

2.5.5.5.7 #define CC_AES_USER_CTX_SIZE_IN_WORDS 131

The size of the user's context prototype (see **CCAesUserContext_t**) in words.

2.5.5.6 typedef documentation

2.5.5.6.1 typedef struct CCAesHwKeyData_t CCAesHwKeyData_t

The AES HW key Data.

2.5.5.6.2 typedef uint8_t CCAesIv_t[CC_AES_IV_SIZE_IN_BYTES]

Defines the IV buffer. A 16-byte array.

2.5.5.6.3 typedef uint8_t CCAesKeyBuffer_t[CC_AES_KEY_MAX_SIZE_IN_BYTES]

Defines the AES key data buffer.

2.5.5.6.4 typedef struct CCAesUserContext_t CCAesUserContext_t

The context prototype of the user.

The argument type that is passed by the user to the AES APIs. The context saves the state of the operation, and must be saved by the user until the end of the API flow.

2.5.5.6.5 typedef struct CCAesUserKeyData_t CCAesUserKeyData_t

The AES key data of the user.

2.5.5.7 Enumeration type documentation

2.5.5.7.1 enum CCAesEncryptMode_t

The AES operation:

- Encrypt.
- Decrypt.

Enumerator:

Enum	Description
CC_AES_ENCRYPT	An AES encrypt operation.
CC_AES_DECRYPT	An AES decrypt operation.
CC_AES_NUM_OF_ENCRYPT_MODES	The maximal number of operations.
CC_AES_ENCRYPT_MODE_LAST	Reserved.

2.5.5.7.2 enum CCAesKeyType_t

The AES key type.

Enumerator:

Enum	Description
CC_AES_USER_KEY	The user key.
CC_AES_PLATFORM_KEY	The Kplt hardware key.
CC_AES_CUSTOMER_KEY	The Kcst hardware key.
CC_AES_NUM_OF_KEY_TYPES	The maximal number of AES key types.
CC_AES_KEY_TYPE_LAST	Reserved.

2.5.5.7.3 enum CCAesOperationMode_t

The AES operation mode.

Enumerator:

Enum	Description
CC_AES_MODE_ECB	ECB mode.

Enum	Description
CC_AES_MODE_CBC	CBC mode.
CC_AES_MODE_CBC_MAC	CBC-MAC mode.
CC_AES_MODE_CTR	CTR mode.
CC_AES_MODE_XCBC_MAC	XCBC-MAC mode.
CC_AES_MODE_CMAC	CMAC mode.
CC_AES_MODE_XTS	XTS mode.
CC_AES_MODE_CBC_CTS	CBC-CTS mode.
CC_AES_MODE_OFB	OFB mode.
CC_AES_MODE_CFB	CFB mode.
CC_AES_NUM_OF_OPERATION_MODES	The maximal number of AES modes.
CC_AES_OPERATION_MODE_LAST	Reserved.

2.5.5.7.4 enum CCAesPaddingType_t

The AES padding type.

Enumerator:

Enum	Description
CC_AES_PADDING_NONE	No padding.
CC_AES_PADDING_PKCS7	PKCS7 padding.
CC_AES_NUM_OF_PADDING_TYPES	The maximal number of AES padding modes.
CC_AES_PADDING_TYPE_LAST	Reserved.

2.5.6 CryptoCell APIs for generation of ECC private and public keys

Contains CryptoCell APIs for generation of ECC private and public keys.

2.5.6.1 Functions

- **CIMPORT_C CCError_t CC_EcpkiKeyPairGenerate** (**CCRndGenerateVectWorkFunc_t** f_rng, void *p_rng, const **CCEcpkiDomain_t** *pDomain, **CCEcpkiUserPrivKey_t** *pUserPrivKey, **CCEcpkiUserPublKey_t** *pUserPublKey, **CCEcpkiKgTempData_t** *pTempData, **CCEcpkiKgCertContext_t** *pFipsCtx)

  Generates a pair of private and public keys in internal representation according to ANSI X9.62-2005: Public Key Cryptography for the Financial Services Industry, The Elliptic Curve Digital Signature Algorithm (ECDSA) standard.

- **CIMPORT_C CCError_t CC_EcpkiKeyPairGenerateBase** (**CCRndGenerateVectWorkFunc_t** f_rng, void *p_rng, const **CCEcpkiDomain_t** *pDomain, const uint32_t *ecX_ptr, const uint32_t *ecY_ptr, **CCEcpkiUserPrivKey_t** *pUserPrivKey, **CCEcpkiUserPublKey_t** *pUserPublKey, **CCEcpkiKgTempData_t** *pTempData, **CCEcpkiKgCertContext_t** *pFipsCtx)

Generates a pair of private and public keys using a configurable base point in internal representation according to ANSI X9.62-2005: Public Key Cryptography for the Financial Services Industry, The Elliptic Curve Digital Signature Algorithm (ECDSA) standard.

2.5.6.2 Function documentation

2.5.6.2.1 CIMPORT_C CCError_t **CC_EcpkiKeyPairGenerate (**CCRndGenerateVectWorkFunc_t ***f_rng*, void * *p_rng*, const** CCEcpkiDomain_t ***pDomain*,** CCEcpkiUserPrivKey_t ***pUserPrivKey*,** CCEcpkiUserPublKey_t ***pUserPublKey*,** CCEcpkiKgTempData_t ***pTempData*,** CCEcpkiKgCertContext_t ***pFipsCtx*)**

Returns:

CC_OK on success.

A non-zero value on failure as defined **cc_ecpki_error.h** or **cc_rnd_error.h**

Parameters:

I/O	Parameter	Description
in	`f_rng`	Pointer to DRBG function
in,out	`p_rng`	Pointer to the random context - the input to f_rng.
in	`pDomain`	Pointer to EC domain (curve).
out	`pUserPrivKey`	Pointer to the private key structure. This structure is used as input to the ECPKI cryptographic primitives.
out	`pUserPublKey`	Pointer to the public key structure. This structure is used as input to the ECPKI cryptographic primitives.
in	`pTempData`	Temporary buffers for internal use, defined in **CCEcpkiKgTempData_t**.
in	`pFipsCtx`	Pointer to temporary buffer used in case FIPS certification if required (may be NULL for all other cases).

2.5.6.2.2 CIMPORT_C CCError_t **CC_EcpkiKeyPairGenerateBase (**CCRndGenerateVectWorkFunc_t ***f_rng*, void * *p_rng*, const** CCEcpkiDomain_t ***pDomain*, const uint32_t * *ecX_ptr*, const uint32_t * *ecY_ptr*,** CCEcpkiUserPrivKey_t ***pUserPrivKey*,** CCEcpkiUserPublKey_t ***pUserPublKey*,** CCEcpkiKgTempData_t ***pTempData*,** CCEcpkiKgCertContext_t ***pFipsCtx*)**

Returns:

CC_OK on success.

A non-zero value on failure as defined **cc_ecpki_error.h** or **cc_rnd_error.h**

Parameters:

I/O	Parameter	Description
in	`f_rng`	Pointer to DRBG function
in,out	`p_rng`	Pointer to the random context - the input to f_rng.

I/O	Parameter	Description
in	pDomain	Pointer to EC domain (curve).
in	ecX_ptr	The X coordinate of the base point.
in	ecY_ptr	The Y coordinate of the base point.
out	pUserPrivKey	Pointer to the private key structure. This structure is used as input to the ECPKI cryptographic primitives.
out	pUserPublKey	Pointer to the public key structure. This structure is used as input to the ECPKI cryptographic primitives.
in	pTempData	Temporary buffers for internal use, defined in **CCEcpkiKgTempData_t**.
in	pFipsCtx	Pointer to temporary buffer used in case FIPS certification if required (may be NULL for all other cases).

2.5.7 CryptoCell ECC APIs

Contains functions and definitions for handling keys used in Elliptic Curves Cryptography (ECC).

2.5.7.1 Modules

- **CryptoCell APIs for generation of ECC private and public keys**

  Contains CryptoCell APIs for generation of ECC private and public keys.

- **CryptoCell ECC specific errors**

  Contains errors that are specific to ECC.

- **CryptoCell ECPKI type definitions**

  Contains CryptoCell ECPKI type definitions.

2.5.7.2 Macros

- #define **CC_EcpkiPubKeyBuild**(pDomain, pPubKeyIn, PublKeySizeInBytes, pUserPublKey) **CC_EcpkiPublKeyBuildAndCheck**((pDomain), (pPubKeyIn), (PublKeySizeInBytes), **CheckPointersAndSizesOnly**, (pUserPublKey), NULL)

  This macro calls **CC_EcpkiPublKeyBuildAndCheck()** function for building the public key while checking input pointers and sizes. For a description of the parameters see **CC_EcpkiPublKeyBuildAndCheck()**.

- #define **CC_EcpkiPubKeyBuildAndPartlyCheck**(pDomain, pPubKeyIn, PublKeySizeInBytes, pUserPublKey, pTempBuff) **CC_EcpkiPublKeyBuildAndCheck**((pDomain), (pPubKeyIn), (PublKeySizeInBytes), **ECpublKeyPartlyCheck**, (pUserPublKey), (pTempBuff))

  This macro calls CC_EcpkiPublKeyBuildAndCheck function for building the public key with partial validation of the key [SEC1] - 3.2.3. For a description of the parameters, see **CC_EcpkiPublKeyBuildAndCheck()**.

- #define **CC_EcpkiPubKeyBuildAndFullCheck**(pDomain, pPubKeyIn, PublKeySizeInBytes, pUserPublKey, pTempBuff) **CC_EcpkiPublKeyBuildAndCheck**((pDomain), (pPubKeyIn), (PublKeySizeInBytes), (**ECpublKeyFullCheck**), (pUserPublKey), (pTempBuff))

  This macro calls CC_EcpkiPublKeyBuildAndCheck function for building the public key with full validation of the key [SEC1] - 3.2.2. For a description of the parameters and return values, see **CC_EcpkiPublKeyBuildAndCheck()**.

2.5.7.3 Functions

- **CIMPORT_C CCError_t CC_EcpkiPrivKeyBuild** (const **CCEcpkiDomain_t** *pDomain, const uint8_t *pPrivKeyIn, size_t PrivKeySizeInBytes, **CCEcpkiUserPrivKey_t** *pUserPrivKey)

  Builds (imports) the user private key structure from an existing private key so that this structure can be used by other EC primitives. This function should be called before using of the private key. Input domain structure must be initialized by EC parameters and auxiliary values, using CC_EcpkiGetDomain() or **CC_EcpkiGetSm2Domain()** functions.

- **CIMPORT_C CCError_t CC_EcpkiPublKeyBuildAndCheck** (const **CCEcpkiDomain_t** *pDomain, uint8_t *pPubKeyIn, size_t PublKeySizeInBytes, **CCEcpkiUserPublKey_t** *pUserPublKey, **CCEcpkiBuildTempData_t** *pTempBuff)

  Builds a user public key structure from an imported public key, so it can be used by other EC primitives. When operating the EC cryptographic algorithms with imported EC public key, this function should be called before using of the public key.

- **CIMPORT_C CCError_t CC_EcpkiPubKeyExport** (**CCEcpkiUserPublKey_t** *pUserPublKey, **CCEcpkiPointCompression_t** compression, uint8_t *pExternPublKey, size_t *pPublKeySizeBytes)

  Converts an existing public key from internal representation to Big-Endian export representation. The function converts the X,Y coordinates of public key EC point to big endianness, and sets the public key, as follows:

  - In case "Uncompressed" point: PubKey = PC||X||Y, PC = 0x4 - single byte;
  - In case of "Hybrid" key PC = 0x6.
  - In case of "Compressed" key PC = 0x2.

2.5.7.4 Function documentation

2.5.7.4.1 CIMPORT_C CCError_t **CC_EcpkiPrivKeyBuild (const** CCEcpkiDomain_t *** *pDomain*, const uint8_t * *pPrivKeyIn*, size_t *PrivKeySizeInBytes*,** CCEcpkiUserPrivKey_t *** *pUserPrivKey*)**

Returns:

CC_OK on success.

A non-zero value on failure as defined **cc_ecpki_error.h**.

Parameters:

I/O	Parameter	Description
in	`pDomain`	The EC domain (curve).
in	`pPrivKeyIn`	Pointer to private key data.
in	`PrivKeySizeInBytes`	Size of private key data (in bytes).
out	`pUserPrivKey`	Pointer to the private key structure. This structure is used as input to the ECPKI cryptographic primitives.

2.5.7.4.2 CIMPORT_C CCError_t **CC_EcpkiPubKeyExport (**CCEcpkiUserPublKey_t ***pUserPublKey**, CCEcpkiPointCompression_t ***compression***, **uint8_t * *pExternPublKey*, size_t * *pPublKeySizeBytes***)

Size of output X and Y coordinates is equal to ModSizeInBytes.

Returns:

`CC_OK` on success.

A non-zero value on failure as defined **cc_ecpki_error.h**.

Parameters:

I/O	Parameter	Description
in	`pUserPublKey`	Pointer to the input public key structure (in Little-Endian form).
in	`compression`	Compression mode: Compressed, Uncompressed or Hybrid.
out	`pExternPublKey`	Pointer to the exported public key array, in compressed or uncompressed or hybrid form: • [PC||X||Y] Big-Endian representation, structured according to [IEEE1363]. • In compressed form, Y is omitted.
in,out	`pPublKeySizeBytes`	Pointer used for the input of the user public key buffer size (in bytes), and the output of the size of the converted public key in bytes.

2.5.7.4.3 CIMPORT_C CCError_t **CC_EcpkiPublKeyBuildAndCheck (const** CCEcpkiDomain_t ***pDomain**, **uint8_t * *pPubKeyIn*, size_t *PublKeySizeInBytes***, CCEcpkiUserPublKey_t ***pUserPublKey**, CCEcpkiBuildTempData_t ***pTempBuff**)

The Incoming public key PublKeyIn structure is big endian bytes array, containing concatenation of PC||X||Y.

PC - point control single byte, defining the type of point: 0x4 - uncompressed, 06,07 - hybrid, 2,3 - compressed.

X,Y - EC point coordinates of public key (y is omitted in compressed form), size of X and Y must be equal to size of EC modulus.

The user may call this function by appropriate macros, according to the necessary validation level in section SEC1. ECC standard: 3.2 of Standards for Efficient Cryptography Group (SECG): SEC1 Elliptic Curve Cryptography and ANSI X9.62-2005: Public Key Cryptography for the Financial Services Industry, The Elliptic Curve Digital Signature Algorithm (ECDSA):

- Checking the input pointers and sizes only - **CC_EcpkiPubKeyBuild()**.
- Partially checking of public key - **CC_EcpkiPubKeyBuildAndPartlyCheck()**.
- Full checking of public key - **CC_EcpkiPubKeyBuildAndFullCheck()**.

Full check mode takes long time and should be used only when it is actually needed.

Returns:

CC_OK on success.

A non-zero value on failure as defined **cc_ecpki_error.h**.

Parameters:

I/O	Parameter	Description
in	pDomain	The EC domain (curve).
in	pPubKeyIn	Pointer to the input public key data, in compressed or uncompressed or hybrid form: [PC||X||Y] Big-Endian representation, structured according to [IEEE1363], where: • X and Y are the public key's EC point coordinates. In compressed form, Y is omitted. • The sizes of X and Y are equal to the size of the EC modulus. • PC is a one-byte point control that defines the type of point compression.
in	PublKeySizeInBytes	The size of public key data (in bytes).
in	pUserPublKey	The required level of public key verification (higher verification level means longer verification time): 0 = Preliminary validation. 1 = Partial validation. 2 = Full validation.
out	ECPublKeyCheckMode_t CheckMode	Pointer to the output public key structure. This structure is used as input to the ECPKI cryptographic primitives.
in	pTempBuff	Pointer for a temporary buffer required for the build function.

2.5.8 CryptoCell ECC specific errors

Contains errors that are specific to ECC.

2.5.8.1 Macros

- #define CC_ECPKI_ILLEGAL_DOMAIN_ID_ERROR (CC_ECPKI_MODULE_ERROR_BASE + 0x1UL)
- #define CC_ECPKI_DOMAIN_PTR_ERROR (CC_ECPKI_MODULE_ERROR_BASE + 0x2UL)
- #define CC_ECPKI_GEN_KEY_INVALID_PRIVATE_KEY_PTR_ERROR (CC_ECPKI_MODULE_ERROR_BASE + 0x3UL)
- #define CC_ECPKI_GEN_KEY_INVALID_PUBLIC_KEY_PTR_ERROR (CC_ECPKI_MODULE_ERROR_BASE + 0x4UL)
- #define CC_ECPKI_GEN_KEY_INVALID_TEMP_DATA_PTR_ERROR (CC_ECPKI_MODULE_ERROR_BASE + 0x5UL)
- #define CC_ECPKI_RND_CONTEXT_PTR_ERROR (CC_ECPKI_MODULE_ERROR_BASE + 0x6UL)
- #define CC_ECPKI_BUILD_KEY_INVALID_COMPRESSION_MODE_ERROR (CC_ECPKI_MODULE_ERROR_BASE + 0x07UL)
- #define CC_ECPKI_BUILD_KEY_ILLEGAL_DOMAIN_ID_ERROR (CC_ECPKI_MODULE_ERROR_BASE + 0x08UL)
- #define CC_ECPKI_BUILD_KEY_INVALID_PRIV_KEY_IN_PTR_ERROR (CC_ECPKI_MODULE_ERROR_BASE + 0x09UL)
- #define CC_ECPKI_BUILD_KEY_INVALID_USER_PRIV_KEY_PTR_ERROR (CC_ECPKI_MODULE_ERROR_BASE + 0x0AUL)
- #define CC_ECPKI_BUILD_KEY_INVALID_PRIV_KEY_SIZE_ERROR (CC_ECPKI_MODULE_ERROR_BASE + 0x0BUL)
- #define CC_ECPKI_BUILD_KEY_INVALID_PRIV_KEY_DATA_ERROR (CC_ECPKI_MODULE_ERROR_BASE + 0x0CUL)
- #define CC_ECPKI_BUILD_KEY_INVALID_PUBL_KEY_IN_PTR_ERROR (CC_ECPKI_MODULE_ERROR_BASE + 0x0DUL)
- #define CC_ECPKI_BUILD_KEY_INVALID_USER_PUBL_KEY_PTR_ERROR (CC_ECPKI_MODULE_ERROR_BASE + 0x0EUL)
- #define CC_ECPKI_BUILD_KEY_INVALID_PUBL_KEY_SIZE_ERROR (CC_ECPKI_MODULE_ERROR_BASE + 0x0FUL)
- #define CC_ECPKI_BUILD_KEY_INVALID_PUBL_KEY_DATA_ERROR (CC_ECPKI_MODULE_ERROR_BASE + 0x10UL)
- #define CC_ECPKI_BUILD_KEY_INVALID_CHECK_MODE_ERROR (CC_ECPKI_MODULE_ERROR_BASE + 0x11UL)
- #define CC_ECPKI_BUILD_KEY_INVALID_TEMP_BUFF_PTR_ERROR (CC_ECPKI_MODULE_ERROR_BASE + 0x12UL)

- #define CC_ECPKI_EXPORT_PUBL_KEY_INVALID_USER_PUBL_KEY_PTR_ERROR (CC_ECPKI_MODULE_ERROR_BASE + 0x14UL)
- #define CC_ECPKI_EXPORT_PUBL_KEY_ILLEGAL_COMPRESSION_MODE_ERROR (CC_ECPKI_MODULE_ERROR_BASE + 0x15UL)
- #define CC_ECPKI_EXPORT_PUBL_KEY_INVALID_EXTERN_PUBL_KEY_PTR_ERROR (CC_ECPKI_MODULE_ERROR_BASE + 0x16UL)
- #define CC_ECPKI_EXPORT_PUBL_KEY_INVALID_PUBL_KEY_SIZE_PTR_ERROR (CC_ECPKI_MODULE_ERROR_BASE + 0x17UL)
- #define CC_ECPKI_EXPORT_PUBL_KEY_INVALID_PUBL_KEY_SIZE_ERROR (CC_ECPKI_MODULE_ERROR_BASE + 0x18UL)
- #define CC_ECPKI_EXPORT_PUBL_KEY_ILLEGAL_DOMAIN_ID_ERROR (CC_ECPKI_MODULE_ERROR_BASE + 0x19UL)
- #define CC_ECPKI_EXPORT_PUBL_KEY_ILLEGAL_VALIDATION_TAG_ERROR (CC_ECPKI_MODULE_ERROR_BASE + 0x1AUL)
- #define CC_ECPKI_EXPORT_PUBL_KEY_INVALID_PUBL_KEY_DATA_ERROR (CC_ECPKI_MODULE_ERROR_BASE + 0x1BUL)
- #define CC_ECPKI_BUILD_DOMAIN_ID_IS_NOT_VALID_ERROR (CC_ECPKI_MODULE_ERROR_BASE + 0x20UL)
- #define CC_ECPKI_BUILD_DOMAIN_DOMAIN_PTR_ERROR (CC_ECPKI_MODULE_ERROR_BASE + 0x21UL)
- #define CC_ECPKI_BUILD_DOMAIN_EC_PARAMETR_PTR_ERROR (CC_ECPKI_MODULE_ERROR_BASE + 0x22UL)
- #define CC_ECPKI_BUILD_DOMAIN_EC_PARAMETR_SIZE_ERROR (CC_ECPKI_MODULE_ERROR_BASE + 0x23UL)
- #define CC_ECPKI_BUILD_DOMAIN_COFACTOR_PARAMS_ERROR (CC_ECPKI_MODULE_ERROR_BASE + 0x24UL)
- #define CC_ECPKI_BUILD_DOMAIN_SECURITY_STRENGTH_ERROR (CC_ECPKI_MODULE_ERROR_BASE + 0x25UL)
- #define CC_ECPKI_BUILD_SCA_RESIST_ILLEGAL_MODE_ERROR (CC_ECPKI_MODULE_ERROR_BASE + 0x26UL)
- #define CC_ECPKI_INTERNAL_ERROR (CC_ECPKI_MODULE_ERROR_BASE + 0x30UL)
- #define CC_ECDH_SVDP_DH_INVALID_PARTNER_PUBL_KEY_PTR_ERROR (CC_ECPKI_MODULE_ERROR_BASE + 0x31UL)
- #define CC_ECDH_SVDP_DH_PARTNER_PUBL_KEY_VALID_TAG_ERROR (CC_ECPKI_MODULE_ERROR_BASE + 0x32UL)
- #define CC_ECDH_SVDP_DH_INVALID_USER_PRIV_KEY_PTR_ERROR (CC_ECPKI_MODULE_ERROR_BASE + 0x33UL)
- #define CC_ECDH_SVDP_DH_USER_PRIV_KEY_VALID_TAG_ERROR (CC_ECPKI_MODULE_ERROR_BASE + 0x34UL)
- #define CC_ECDH_SVDP_DH_INVALID_SHARED_SECRET_VALUE_PTR_ERROR (CC_ECPKI_MODULE_ERROR_BASE + 0x35UL)

- #define CC_ECDH_SVDP_DH_INVALID_TEMP_DATA_PTR_ERROR (CC_ECPKI_MODULE_ERROR_BASE + 0x36UL)
- #define CC_ECDH_SVDP_DH_INVALID_SHARED_SECRET_VALUE_SIZE_PTR_ERROR (CC_ECPKI_MODULE_ERROR_BASE + 0x37UL)
- #define CC_ECDH_SVDP_DH_INVALID_SHARED_SECRET_VALUE_SIZE_ERROR (CC_ECPKI_MODULE_ERROR_BASE + 0x38UL)
- #define CC_ECDH_SVDP_DH_ILLEGAL_DOMAIN_ID_ERROR (CC_ECPKI_MODULE_ERROR_BASE + 0x39UL)
- #define CC_ECDH_SVDP_DH_NOT_CONCENT_PUBL_AND_PRIV_DOMAIN_ID_ERROR (CC_ECPKI_MODULE_ERROR_BASE + 0x3AUL)
- #define CC_ECDSA_SIGN_INVALID_DOMAIN_ID_ERROR (CC_ECPKI_MODULE_ERROR_BASE + 0x50UL)
- #define CC_ECDSA_SIGN_INVALID_USER_CONTEXT_PTR_ERROR (CC_ECPKI_MODULE_ERROR_BASE + 0x51UL)
- #define CC_ECDSA_SIGN_INVALID_USER_PRIV_KEY_PTR_ERROR (CC_ECPKI_MODULE_ERROR_BASE + 0x52UL)
- #define CC_ECDSA_SIGN_ILLEGAL_HASH_OP_MODE_ERROR (CC_ECPKI_MODULE_ERROR_BASE + 0x53UL)
- #define CC_ECDSA_SIGN_INVALID_MESSAGE_DATA_IN_PTR_ERROR (CC_ECPKI_MODULE_ERROR_BASE + 0x54UL)
- #define CC_ECDSA_SIGN_INVALID_MESSAGE_DATA_IN_SIZE_ERROR (CC_ECPKI_MODULE_ERROR_BASE + 0x55UL)
- #define CC_ECDSA_SIGN_USER_CONTEXT_VALIDATION_TAG_ERROR (CC_ECPKI_MODULE_ERROR_BASE + 0x57UL)
- #define CC_ECDSA_SIGN_USER_PRIV_KEY_VALIDATION_TAG_ERROR (CC_ECPKI_MODULE_ERROR_BASE + 0x58UL)
- #define CC_ECDSA_SIGN_INVALID_SIGNATURE_OUT_PTR_ERROR (CC_ECPKI_MODULE_ERROR_BASE + 0x60UL)
- #define CC_ECDSA_SIGN_INVALID_SIGNATURE_OUT_SIZE_PTR_ERROR (CC_ECPKI_MODULE_ERROR_BASE + 0x61UL)
- #define CC_ECDSA_SIGN_INVALID_SIGNATURE_OUT_SIZE_ERROR (CC_ECPKI_MODULE_ERROR_BASE + 0x62UL)
- #define CC_ECDSA_SIGN_INVALID_IS_EPHEMER_KEY_INTERNAL_ERROR (CC_ECPKI_MODULE_ERROR_BASE + 0x63UL)
- #define CC_ECDSA_SIGN_INVALID_EPHEMERAL_KEY_PTR_ERROR (CC_ECPKI_MODULE_ERROR_BASE + 0x64UL)
- #define CC_ECDSA_SIGN_INVALID_RND_CONTEXT_PTR_ERROR (CC_ECPKI_MODULE_ERROR_BASE + 0x65UL)
- #define CC_ECDSA_SIGN_INVALID_RND_FUNCTION_PTR_ERROR (CC_ECPKI_MODULE_ERROR_BASE + 0x66UL)

- #define CC_ECDSA_SIGN_SIGNING_ERROR (CC_ECPKI_MODULE_ERROR_BASE + 0x67UL)
- #define CC_ECDSA_VERIFY_INVALID_DOMAIN_ID_ERROR (CC_ECPKI_MODULE_ERROR_BASE + 0x70UL)
- #define CC_ECDSA_VERIFY_INVALID_USER_CONTEXT_PTR_ERROR (CC_ECPKI_MODULE_ERROR_BASE + 0x71UL)
- #define CC_ECDSA_VERIFY_INVALID_SIGNER_PUBL_KEY_PTR_ERROR (CC_ECPKI_MODULE_ERROR_BASE + 0x72UL)
- #define CC_ECDSA_VERIFY_ILLEGAL_HASH_OP_MODE_ERROR (CC_ECPKI_MODULE_ERROR_BASE + 0x73UL)
- #define CC_ECDSA_VERIFY_INVALID_SIGNATURE_IN_PTR_ERROR (CC_ECPKI_MODULE_ERROR_BASE + 0x76UL)
- #define CC_ECDSA_VERIFY_INVALID_SIGNATURE_SIZE_ERROR (CC_ECPKI_MODULE_ERROR_BASE + 0x77UL)
- #define CC_ECDSA_VERIFY_INVALID_MESSAGE_DATA_IN_PTR_ERROR (CC_ECPKI_MODULE_ERROR_BASE + 0x80UL)
- #define CC_ECDSA_VERIFY_INVALID_MESSAGE_DATA_IN_SIZE_ERROR (CC_ECPKI_MODULE_ERROR_BASE + 0x81UL)
- #define CC_ECDSA_VERIFY_USER_CONTEXT_VALIDATION_TAG_ERROR (CC_ECPKI_MODULE_ERROR_BASE + 0x82UL)
- #define CC_ECDSA_VERIFY_SIGNER_PUBL_KEY_VALIDATION_TAG_ERROR (CC_ECPKI_MODULE_ERROR_BASE + 0x83UL)
- #define CC_ECDSA_VERIFY_INCONSISTENT_VERIFY_ERROR (CC_ECPKI_MODULE_ERROR_BASE + 0x84UL)
- #define CC_ECC_ILLEGAL_HASH_MODE_ERROR (CC_ECPKI_MODULE_ERROR_BASE + 0x85UL)
- #define CC_ECPKI_INVALID_RND_FUNC_PTR_ERROR (CC_ECPKI_MODULE_ERROR_BASE + 0x90UL)
- #define CC_ECPKI_INVALID_RND_CTX_PTR_ERROR (CC_ECPKI_MODULE_ERROR_BASE + 0x91UL)
- #define CC_ECPKI_INVALID_DOMAIN_ID_ERROR (CC_ECPKI_MODULE_ERROR_BASE + 0x92UL)
- #define CC_ECPKI_INVALID_PRIV_KEY_TAG_ERROR (CC_ECPKI_MODULE_ERROR_BASE + 0x93UL)
- #define CC_ECPKI_INVALID_PUBL_KEY_TAG_ERROR (CC_ECPKI_MODULE_ERROR_BASE + 0x94UL)
- #define CC_ECPKI_INVALID_DATA_IN_PASSED_STRUCT_ERROR (CC_ECPKI_MODULE_ERROR_BASE + 0x95UL)
- #define CC_ECPKI_INVALID_BASE_POINT_PTR_ERROR (CC_ECPKI_MODULE_ERROR_BASE + 0x96UL)

- #define CC_ECPKI_INVALID_OUT_HASH_PTR_ERROR (CC_ECPKI_MODULE_ERROR_BASE + 0x97UL)
- #define CC_ECPKI_INVALID_OUT_HASH_SIZE_ERROR (CC_ECPKI_MODULE_ERROR_BASE + 0x98UL)
- #define CC_ECPKI_INVALID_IN_HASH_PTR_ERROR (CC_ECPKI_MODULE_ERROR_BASE + 0x99UL)
- #define CC_ECPKI_INVALID_IN_HASH_SIZE_ERROR (CC_ECPKI_MODULE_ERROR_BASE + 0x9AUL)
- #define CC_ECPKI_SM2_INVALID_KE_CONTEXT_PTR (CC_ECPKI_MODULE_ERROR_BASE + 0xA0UL)
- #define CC_ECPKI_SM2_INVALID_ID_PTR (CC_ECPKI_MODULE_ERROR_BASE + 0xA1UL)
- #define CC_ECPKI_SM2_INVALID_ID_SIZE (CC_ECPKI_MODULE_ERROR_BASE + 0xA2UL)
- #define CC_ECPKI_SM2_INVALID_IN_PARAM_SIZE (CC_ECPKI_MODULE_ERROR_BASE + 0xA3UL)
- #define CC_ECPKI_SM2_INVALID_OUT_PARAM_SIZE (CC_ECPKI_MODULE_ERROR_BASE + 0xA4UL)
- #define CC_ECPKI_SM2_INVALID_OUT_PARAM_PTR (CC_ECPKI_MODULE_ERROR_BASE + 0xA5UL)
- #define CC_ECPKI_SM2_INVALID_CONTEXT (CC_ECPKI_MODULE_ERROR_BASE + 0xA6UL)
- #define CC_ECPKI_SM2_INVALID_EPHEMERAL_PUB_IN_PTR (CC_ECPKI_MODULE_ERROR_BASE + 0xA7UL)
- #define CC_ECPKI_SM2_INVALID_EPHEMERAL_PUB_OUT_PTR (CC_ECPKI_MODULE_ERROR_BASE + 0xA8UL)
- #define CC_ECPKI_SM2_INVALID_SHARED_SECRET_OUT_PTR (CC_ECPKI_MODULE_ERROR_BASE + 0xA9UL)
- #define CC_ECPKI_SM2_INVALID_SHARED_SECRET_IN_PTR (CC_ECPKI_MODULE_ERROR_BASE + 0xAAUL)
- #define CC_ECPKI_SM2_INVALID_IN_PARAM_PTR (CC_ECPKI_MODULE_ERROR_BASE + 0xABUL)
- #define CC_ECPKI_SM2_INVALID_EPHEMERAL_PRIV_IN_PTR (CC_ECPKI_MODULE_ERROR_BASE + 0xACUL)
- #define CC_ECPKI_SM2_CONFIRMATION_FAILED (CC_ECPKI_MODULE_ERROR_BASE + 0xADUL)
- #define CC_ECIES_INVALID_PUBL_KEY_PTR_ERROR (CC_ECPKI_MODULE_ERROR_BASE + 0xE0UL)
- #define CC_ECIES_INVALID_PUBL_KEY_TAG_ERROR (CC_ECPKI_MODULE_ERROR_BASE + 0xE1UL)
- #define CC_ECIES_INVALID_PRIV_KEY_PTR_ERROR (CC_ECPKI_MODULE_ERROR_BASE + 0xE2UL)

- #define CC_ECIES_INVALID_PRIV_KEY_TAG_ERROR (CC_ECPKI_MODULE_ERROR_BASE + 0xE3UL)
- #define CC_ECIES_INVALID_PRIV_KEY_VALUE_ERROR (CC_ECPKI_MODULE_ERROR_BASE + 0xE4UL)
- #define CC_ECIES_INVALID_KDF_DERIV_MODE_ERROR (CC_ECPKI_MODULE_ERROR_BASE + 0xE5UL)
- #define CC_ECIES_INVALID_KDF_HASH_MODE_ERROR (CC_ECPKI_MODULE_ERROR_BASE + 0xE6UL)
- #define CC_ECIES_INVALID_SECRET_KEY_PTR_ERROR (CC_ECPKI_MODULE_ERROR_BASE + 0xE7UL)
- #define CC_ECIES_INVALID_SECRET_KEY_SIZE_ERROR (CC_ECPKI_MODULE_ERROR_BASE + 0xE8UL)
- #define CC_ECIES_INVALID_CIPHER_DATA_PTR_ERROR (CC_ECPKI_MODULE_ERROR_BASE + 0xE9UL)
- #define CC_ECIES_INVALID_CIPHER_DATA_SIZE_PTR_ERROR (CC_ECPKI_MODULE_ERROR_BASE + 0xEAUL)
- #define CC_ECIES_INVALID_CIPHER_DATA_SIZE_ERROR (CC_ECPKI_MODULE_ERROR_BASE + 0xEBUL)
- #define CC_ECIES_INVALID_TEMP_DATA_PTR_ERROR (CC_ECPKI_MODULE_ERROR_BASE + 0xECUL)
- #define CC_ECIES_INVALID_TEMP_DATA_SIZE_ERROR (CC_ECPKI_MODULE_ERROR_BASE + 0xEDUL)
- #define CC_ECIES_INVALID_EPHEM_KEY_PAIR_PTR_ERROR (CC_ECPKI_MODULE_ERROR_BASE + 0xEEUL)
- #define CC_ECIES_INVALID_PTR (CC_ECPKI_MODULE_ERROR_BASE + 0xEFUL)

2.5.8.2 Macro definition documentation

2.5.8.2.1 #define CC_ECC_ILLEGAL_HASH_MODE_ERROR (CC_ECPKI_MODULE_ERROR_BASE + 0x85UL)

Illegal hash mode.

2.5.8.2.2 #define CC_ECDH_SVDP_DH_ILLEGAL_DOMAIN_ID_ERROR (CC_ECPKI_MODULE_ERROR_BASE + 0x39UL)

Illegal domain ID.

2.5.8.2.3 #define CC_ECDH_SVDP_DH_INVALID_PARTNER_PUBL_KEY_PTR_ERROR (CC_ECPKI_MODULE_ERROR_BASE + 0x31UL)

Illegal partner's public key pointer.

2.5.8.2.4 #define CC_ECDH_SVDP_DH_INVALID_SHARED_SECRET_VALUE_PTR_ERROR (CC_ECPKI_MODULE_ERROR_BASE + 0x35UL)

Illegal shared secret pointer.

2.5.8.2.5 #define CC_ECDH_SVDP_DH_INVALID_SHARED_SECRET_VALUE_SIZE_ERROR (CC_ECPKI_MODULE_ERROR_BASE + 0x38UL)

Illegal shared secret size.

2.5.8.2.6 #define CC_ECDH_SVDP_DH_INVALID_SHARED_SECRET_VALUE_SIZE_PTR_ERROR (CC_ECPKI_MODULE_ERROR_BASE + 0x37UL)

Illegal shared secret size pointer.

2.5.8.2.7 #define CC_ECDH_SVDP_DH_INVALID_TEMP_DATA_PTR_ERROR (CC_ECPKI_MODULE_ERROR_BASE + 0x36UL)

Illegal temporary buffer pointer.

2.5.8.2.8 #define CC_ECDH_SVDP_DH_INVALID_USER_PRIV_KEY_PTR_ERROR (CC_ECPKI_MODULE_ERROR_BASE + 0x33UL)

Illegal user private key pointer.

2.5.8.2.9 #define CC_ECDH_SVDP_DH_NOT_CONCENT_PUBL_AND_PRIV_DOMAIN_ID_ERROR (CC_ECPKI_MODULE_ERROR_BASE + 0x3AUL)

Illegal private and public domain ID are different.

2.5.8.2.10 #define CC_ECDH_SVDP_DH_PARTNER_PUBL_KEY_VALID_TAG_ERROR (CC_ECPKI_MODULE_ERROR_BASE + 0x32UL)

Partner's public key validation failed.

2.5.8.2.11 #define CC_ECDH_SVDP_DH_USER_PRIV_KEY_VALID_TAG_ERROR (CC_ECPKI_MODULE_ERROR_BASE + 0x34UL)

Private key validation failed.

2.5.8.2.12 #define CC_ECDSA_SIGN_ILLEGAL_HASH_OP_MODE_ERROR (CC_ECPKI_MODULE_ERROR_BASE + 0x53UL)

Illegal hash operation mode.

2.5.8.2.13 #define CC_ECDSA_SIGN_INVALID_DOMAIN_ID_ERROR (CC_ECPKI_MODULE_ERROR_BASE + 0x50UL)

Illegal domain ID.

2.5.8.2.14 #define CC_ECDSA_SIGN_INVALID_EPHEMERAL_KEY_PTR_ERROR (CC_ECPKI_MODULE_ERROR_BASE + 0x64UL)

Illegal ephemeral key pointer.

2.5.8.2.15 #define CC_ECDSA_SIGN_INVALID_IS_EPHEMER_KEY_INTERNAL_ERROR (CC_ECPKI_MODULE_ERROR_BASE + 0x63UL)

Ephemeral key error.

2.5.8.2.16 #define CC_ECDSA_SIGN_INVALID_MESSAGE_DATA_IN_PTR_ERROR (CC_ECPKI_MODULE_ERROR_BASE + 0x54UL)

Illegal data in pointer.

2.5.8.2.17 #define CC_ECDSA_SIGN_INVALID_MESSAGE_DATA_IN_SIZE_ERROR (CC_ECPKI_MODULE_ERROR_BASE + 0x55UL)

Illegal data in size.

2.5.8.2.18 #define CC_ECDSA_SIGN_INVALID_RND_CONTEXT_PTR_ERROR (CC_ECPKI_MODULE_ERROR_BASE + 0x65UL)

Illegal RND context pointer.

2.5.8.2.19 #define CC_ECDSA_SIGN_INVALID_RND_FUNCTION_PTR_ERROR (CC_ECPKI_MODULE_ERROR_BASE + 0x66UL)

Illegal RND function pointer.

2.5.8.2.20 #define CC_ECDSA_SIGN_INVALID_SIGNATURE_OUT_PTR_ERROR (CC_ECPKI_MODULE_ERROR_BASE + 0x60UL)

Illegal signature pointer.

2.5.8.2.21 #define CC_ECDSA_SIGN_INVALID_SIGNATURE_OUT_SIZE_ERROR (CC_ECPKI_MODULE_ERROR_BASE + 0x62UL)

Illegal signature size.

2.5.8.2.22 #define CC_ECDSA_SIGN_INVALID_SIGNATURE_OUT_SIZE_PTR_ERROR (CC_ECPKI_MODULE_ERROR_BASE + 0x61UL)

Illegal signature size pointer.

2.5.8.2.23 #define CC_ECDSA_SIGN_INVALID_USER_CONTEXT_PTR_ERROR (CC_ECPKI_MODULE_ERROR_BASE + 0x51UL)

Illegal context pointer.

2.5.8.2.24 #define CC_ECDSA_SIGN_INVALID_USER_PRIV_KEY_PTR_ERROR (CC_ECPKI_MODULE_ERROR_BASE + 0x52UL)

Illegal private key pointer.

2.5.8.2.25 #define CC_ECDSA_SIGN_SIGNING_ERROR (CC_ECPKI_MODULE_ERROR_BASE + 0x67UL)

Signature calculation failed.

2.5.8.2.26 #define CC_ECDSA_SIGN_USER_CONTEXT_VALIDATION_TAG_ERROR (CC_ECPKI_MODULE_ERROR_BASE + 0x57UL)

Context validation failed.

2.5.8.2.27 #define CC_ECDSA_SIGN_USER_PRIV_KEY_VALIDATION_TAG_ERROR (CC_ECPKI_MODULE_ERROR_BASE + 0x58UL)

User private key validation failed.

2.5.8.2.28 #define CC_ECDSA_VERIFY_ILLEGAL_HASH_OP_MODE_ERROR (CC_ECPKI_MODULE_ERROR_BASE + 0x73UL)

Illegal hash operation mode.

2.5.8.2.29 #define CC_ECDSA_VERIFY_INCONSISTENT_VERIFY_ERROR (CC_ECPKI_MODULE_ERROR_BASE + 0x84UL)

Verification failed.

2.5.8.2.30 #define CC_ECDSA_VERIFY_INVALID_DOMAIN_ID_ERROR (CC_ECPKI_MODULE_ERROR_BASE + 0x70UL)

Illegal domain ID.

2.5.8.2.31 #define CC_ECDSA_VERIFY_INVALID_MESSAGE_DATA_IN_PTR_ERROR (CC_ECPKI_MODULE_ERROR_BASE + 0x80UL)

Illegal data in pointer.

2.5.8.2.32 #define CC_ECDSA_VERIFY_INVALID_MESSAGE_DATA_IN_SIZE_ERROR (CC_ECPKI_MODULE_ERROR_BASE + 0x81UL)

Illegal data in size.

2.5.8.2.33 #define CC_ECDSA_VERIFY_INVALID_SIGNATURE_IN_PTR_ERROR (CC_ECPKI_MODULE_ERROR_BASE + 0x76UL)

Illegal signature pointer.

2.5.8.2.34 #define CC_ECDSA_VERIFY_INVALID_SIGNATURE_SIZE_ERROR (CC_ECPKI_MODULE_ERROR_BASE + 0x77UL)

Illegal signature size.

2.5.8.2.35 #define CC_ECDSA_VERIFY_INVALID_SIGNER_PUBL_KEY_PTR_ERROR (CC_ECPKI_MODULE_ERROR_BASE + 0x72UL)

Illegal public key pointer.

2.5.8.2.36 #define CC_ECDSA_VERIFY_INVALID_USER_CONTEXT_PTR_ERROR (CC_ECPKI_MODULE_ERROR_BASE + 0x71UL)

Illegal user context pointer.

2.5.8.2.37 #define CC_ECDSA_VERIFY_SIGNER_PUBL_KEY_VALIDATION_TAG_ERROR (CC_ECPKI_MODULE_ERROR_BASE + 0x83UL)

Public key validation failed.

2.5.8.2.38 #define CC_ECDSA_VERIFY_USER_CONTEXT_VALIDATION_TAG_ERROR (CC_ECPKI_MODULE_ERROR_BASE + 0x82UL)

Context validation failed.

2.5.8.2.39 #define CC_ECIES_INVALID_CIPHER_DATA_PTR_ERROR (CC_ECPKI_MODULE_ERROR_BASE + 0xE9UL)

Illegal cipher data pointer.

2.5.8.2.40 #define CC_ECIES_INVALID_CIPHER_DATA_SIZE_ERROR (CC_ECPKI_MODULE_ERROR_BASE + 0xEBUL)

Illegal cipher data size.

2.5.8.2.41 #define CC_ECIES_INVALID_CIPHER_DATA_SIZE_PTR_ERROR (CC_ECPKI_MODULE_ERROR_BASE + 0xEAUL)

Illegal cipher data size pointer.

2.5.8.2.42 #define CC_ECIES_INVALID_EPHEM_KEY_PAIR_PTR_ERROR (CC_ECPKI_MODULE_ERROR_BASE + 0xEEUL)

Illegal ephemeral key pointer

2.5.8.2.43 #define CC_ECIES_INVALID_KDF_DERIV_MODE_ERROR (CC_ECPKI_MODULE_ERROR_BASE + 0xE5UL)

Illegal KDF derivation mode.

2.5.8.2.44 #define CC_ECIES_INVALID_KDF_HASH_MODE_ERROR (CC_ECPKI_MODULE_ERROR_BASE + 0xE6UL)

Illegal KDF hash mode.

2.5.8.2.45 #define CC_ECIES_INVALID_PRIV_KEY_PTR_ERROR (CC_ECPKI_MODULE_ERROR_BASE + 0xE2UL)

Illegal private key pointer.

2.5.8.2.46 #define CC_ECIES_INVALID_PRIV_KEY_TAG_ERROR (CC_ECPKI_MODULE_ERROR_BASE + 0xE3UL)

Private key validation failed.

2.5.8.2.47 #define CC_ECIES_INVALID_PRIV_KEY_VALUE_ERROR (CC_ECPKI_MODULE_ERROR_BASE + 0xE4UL)

Illegal private key value.

2.5.8.2.48 #define CC_ECIES_INVALID_PTR (CC_ECPKI_MODULE_ERROR_BASE + 0xEFUL)

NULL pointer

2.5.8.2.49 #define CC_ECIES_INVALID_PUBL_KEY_PTR_ERROR (CC_ECPKI_MODULE_ERROR_BASE + 0xE0UL)

Illegal public key pointer.

2.5.8.2.50 #define CC_ECIES_INVALID_PUBL_KEY_TAG_ERROR (CC_ECPKI_MODULE_ERROR_BASE + 0xE1UL)

Public key validation failed.

2.5.8.2.51 #define CC_ECIES_INVALID_SECRET_KEY_PTR_ERROR (CC_ECPKI_MODULE_ERROR_BASE + 0xE7UL)

Illegal secret key pointer.

2.5.8.2.52 #define CC_ECIES_INVALID_SECRET_KEY_SIZE_ERROR (CC_ECPKI_MODULE_ERROR_BASE + 0xE8UL)

Illegal secret key size.

2.5.8.2.53 #define CC_ECIES_INVALID_TEMP_DATA_PTR_ERROR (CC_ECPKI_MODULE_ERROR_BASE + 0xECUL)

Illegal temporary buffer pointer.

2.5.8.2.54 #define CC_ECIES_INVALID_TEMP_DATA_SIZE_ERROR (CC_ECPKI_MODULE_ERROR_BASE + 0xEDUL)

Illegal temporary buffer size

2.5.8.2.55 #define CC_ECPKI_BUILD_DOMAIN_COFACTOR_PARAMS_ERROR (CC_ECPKI_MODULE_ERROR_BASE + 0x24UL)

Illegal domain cofactor parameters.

2.5.8.2.56 #define CC_ECPKI_BUILD_DOMAIN_DOMAIN_PTR_ERROR (CC_ECPKI_MODULE_ERROR_BASE + 0x21UL)

Illegal domain ID pointer.

2.5.8.2.57 #define CC_ECPKI_BUILD_DOMAIN_EC_PARAMETR_PTR_ERROR (CC_ECPKI_MODULE_ERROR_BASE + 0x22UL)

Illegal domain parameter pointer.

2.5.8.2.58 #define CC_ECPKI_BUILD_DOMAIN_EC_PARAMETR_SIZE_ERROR (CC_ECPKI_MODULE_ERROR_BASE + 0x23UL)

Illegal domain parameter size.

2.5.8.2.59 #define CC_ECPKI_BUILD_DOMAIN_ID_IS_NOT_VALID_ERROR (CC_ECPKI_MODULE_ERROR_BASE + 0x20UL)

Illegal domain ID.

2.5.8.2.60 #define CC_ECPKI_BUILD_DOMAIN_SECURITY_STRENGTH_ERROR (CC_ECPKI_MODULE_ERROR_BASE + 0x25UL)

Insufficient strength.

2.5.8.2.61 #define CC_ECPKI_BUILD_KEY_ILLEGAL_DOMAIN_ID_ERROR (CC_ECPKI_MODULE_ERROR_BASE + 0x08UL)

Illegal domain ID.

2.5.8.2.62 #define CC_ECPKI_BUILD_KEY_INVALID_CHECK_MODE_ERROR (CC_ECPKI_MODULE_ERROR_BASE + 0x11UL)

Illegal EC build check mode option.

2.5.8.2.63 #define CC_ECPKI_BUILD_KEY_INVALID_COMPRESSION_MODE_ERROR (CC_ECPKI_MODULE_ERROR_BASE + 0x07UL)

Illegal compression mode.

2.5.8.2.64 #define CC_ECPKI_BUILD_KEY_INVALID_PRIV_KEY_DATA_ERROR (CC_ECPKI_MODULE_ERROR_BASE + 0x0CUL)

Illegal private key data.

2.5.8.2.65 #define CC_ECPKI_BUILD_KEY_INVALID_PRIV_KEY_IN_PTR_ERROR (CC_ECPKI_MODULE_ERROR_BASE + 0x09UL)

Illegal private key pointer.

2.5.8.2.66 #define CC_ECPKI_BUILD_KEY_INVALID_PRIV_KEY_SIZE_ERROR (CC_ECPKI_MODULE_ERROR_BASE + 0x0BUL)

Illegal private key size.

2.5.8.2.67 #define CC_ECPKI_BUILD_KEY_INVALID_PUBL_KEY_DATA_ERROR (CC_ECPKI_MODULE_ERROR_BASE + 0x10UL)

Illegal public key data.

2.5.8.2.68 #define CC_ECPKI_BUILD_KEY_INVALID_PUBL_KEY_IN_PTR_ERROR (CC_ECPKI_MODULE_ERROR_BASE + 0x0DUL)

Illegal public key pointer.

2.5.8.2.69 #define CC_ECPKI_BUILD_KEY_INVALID_PUBL_KEY_SIZE_ERROR (CC_ECPKI_MODULE_ERROR_BASE + 0x0FUL)

Illegal public key size.

2.5.8.2.70 #define CC_ECPKI_BUILD_KEY_INVALID_TEMP_BUFF_PTR_ERROR (CC_ECPKI_MODULE_ERROR_BASE + 0x12UL)

Illegal temporary buffer pointer.

2.5.8.2.71 #define CC_ECPKI_BUILD_KEY_INVALID_USER_PRIV_KEY_PTR_ERROR (CC_ECPKI_MODULE_ERROR_BASE + 0x0AUL)

Illegal private key structure pointer.

2.5.8.2.72 #define CC_ECPKI_BUILD_KEY_INVALID_USER_PUBL_KEY_PTR_ERROR (CC_ECPKI_MODULE_ERROR_BASE + 0x0EUL)

Illegal public key structure pointer.

2.5.8.2.73 #define CC_ECPKI_BUILD_SCA_RESIST_ILLEGAL_MODE_ERROR (CC_ECPKI_MODULE_ERROR_BASE + 0x26UL)

SCA resistance error.

2.5.8.2.74 #define CC_ECPKI_DOMAIN_PTR_ERROR (CC_ECPKI_MODULE_ERROR_BASE + 0x2UL)

Illegal domain pointer.

2.5.8.2.75 #define CC_ECPKI_EXPORT_PUBL_KEY_ILLEGAL_COMPRESSION_MODE_ERROR (CC_ECPKI_MODULE_ERROR_BASE + 0x15UL)

Illegal public key compression mode.

2.5.8.2.76 #define CC_ECPKI_EXPORT_PUBL_KEY_ILLEGAL_DOMAIN_ID_ERROR (CC_ECPKI_MODULE_ERROR_BASE + 0x19UL)

Illegal domain ID.

2.5.8.2.77 #define CC_ECPKI_EXPORT_PUBL_KEY_ILLEGAL_VALIDATION_TAG_ERROR (CC_ECPKI_MODULE_ERROR_BASE + 0x1AUL)

Validation of public key failed.

2.5.8.2.78 #define CC_ECPKI_EXPORT_PUBL_KEY_INVALID_EXTERN_PUBL_KEY_PTR_ERROR (CC_ECPKI_MODULE_ERROR_BASE + 0x16UL)

Illegal output public key pointer.

2.5.8.2.79 #define CC_ECPKI_EXPORT_PUBL_KEY_INVALID_PUBL_KEY_DATA_ERROR (CC_ECPKI_MODULE_ERROR_BASE + 0x1BUL)

Validation of public key failed.

2.5.8.2.80 #define CC_ECPKI_EXPORT_PUBL_KEY_INVALID_PUBL_KEY_SIZE_ERROR (CC_ECPKI_MODULE_ERROR_BASE + 0x18UL)

Illegal output public key size.

2.5.8.2.81 #define CC_ECPKI_EXPORT_PUBL_KEY_INVALID_PUBL_KEY_SIZE_PTR_ERROR (CC_ECPKI_MODULE_ERROR_BASE + 0x17UL)

Illegal output public key size pointer.

2.5.8.2.82 #define CC_ECPKI_EXPORT_PUBL_KEY_INVALID_USER_PUBL_KEY_PTR_ERROR (CC_ECPKI_MODULE_ERROR_BASE + 0x14UL)

Illegal public key structure pointer.

2.5.8.2.83 #define CC_ECPKI_GEN_KEY_INVALID_PRIVATE_KEY_PTR_ERROR (CC_ECPKI_MODULE_ERROR_BASE + 0x3UL)

The CryptoCell ECPKI GEN KEY PAIR module errors

Illegal private key pointer.

2.5.8.2.84 #define CC_ECPKI_GEN_KEY_INVALID_PUBLIC_KEY_PTR_ERROR (CC_ECPKI_MODULE_ERROR_BASE + 0x4UL)

Illegal public key pointer.

2.5.8.2.85 #define CC_ECPKI_GEN_KEY_INVALID_TEMP_DATA_PTR_ERROR (CC_ECPKI_MODULE_ERROR_BASE + 0x5UL)

Illegal temporary buffer pointer.

2.5.8.2.86 #define CC_ECPKI_ILLEGAL_DOMAIN_ID_ERROR (CC_ECPKI_MODULE_ERROR_BASE + 0x1UL)

Illegal domain ID.

2.5.8.2.87 #define CC_ECPKI_INTERNAL_ERROR (CC_ECPKI_MODULE_ERROR_BASE + 0x30UL)

Internal error

2.5.8.2.88 #define CC_ECPKI_INVALID_BASE_POINT_PTR_ERROR (CC_ECPKI_MODULE_ERROR_BASE + 0x96UL)

Illegal Base point pointer.

2.5.8.2.89 #define CC_ECPKI_INVALID_DATA_IN_PASSED_STRUCT_ERROR (CC_ECPKI_MODULE_ERROR_BASE + 0x95UL)

Illegal data in.

2.5.8.2.90 #define CC_ECPKI_INVALID_DOMAIN_ID_ERROR (CC_ECPKI_MODULE_ERROR_BASE + 0x92UL)

Illegal domain ID.

2.5.8.2.91 #define CC_ECPKI_INVALID_IN_HASH_PTR_ERROR (CC_ECPKI_MODULE_ERROR_BASE + 0x99UL)

Illegal in hash pointer.

2.5.8.2.92 #define CC_ECPKI_INVALID_IN_HASH_SIZE_ERROR (CC_ECPKI_MODULE_ERROR_BASE + 0x9AUL)

Illegal in hash length.

2.5.8.2.93 #define CC_ECPKI_INVALID_OUT_HASH_PTR_ERROR (CC_ECPKI_MODULE_ERROR_BASE + 0x97UL)

Illegal out hash pointer.

2.5.8.2.94 #define CC_ECPKI_INVALID_OUT_HASH_SIZE_ERROR (CC_ECPKI_MODULE_ERROR_BASE + 0x98UL)

Illegal out hash length.

2.5.8.2.95 #define CC_ECPKI_INVALID_PRIV_KEY_TAG_ERROR (CC_ECPKI_MODULE_ERROR_BASE + 0x93UL)

Private key validation failed.

2.5.8.2.96 #define CC_ECPKI_INVALID_PUBL_KEY_TAG_ERROR (CC_ECPKI_MODULE_ERROR_BASE + 0x94UL)

Public key validation failed.

2.5.8.2.97 #define CC_ECPKI_INVALID_RND_CTX_PTR_ERROR (CC_ECPKI_MODULE_ERROR_BASE + 0x91UL)

Illegal RND context pointer.

2.5.8.2.98 #define CC_ECPKI_INVALID_RND_FUNC_PTR_ERROR (CC_ECPKI_MODULE_ERROR_BASE + 0x90UL)

Illegal RND function pointer.

2.5.8.2.99 #define CC_ECPKI_RND_CONTEXT_PTR_ERROR (CC_ECPKI_MODULE_ERROR_BASE + 0x6UL)

Illegal RND context pointer.

2.5.8.2.100 #define CC_ECPKI_SM2_CONFIRMATION_FAILED (CC_ECPKI_MODULE_ERROR_BASE + 0xADUL)

The SM2 confirmation failed. The other party's confirmation value is different than the confirmation value calculated.

2.5.8.2.101 #define CC_ECPKI_SM2_INVALID_CONTEXT (CC_ECPKI_MODULE_ERROR_BASE + 0xA6UL)

Illegal key context.

2.5.8.2.102 #define CC_ECPKI_SM2_INVALID_EPHEMERAL_PRIV_IN_PTR (CC_ECPKI_MODULE_ERROR_BASE + 0xACUL)

Illegal key in parameter pointer.

2.5.8.2.103 #define CC_ECPKI_SM2_INVALID_EPHEMERAL_PUB_IN_PTR (CC_ECPKI_MODULE_ERROR_BASE + 0xA7UL)

Illegal ephemeral public key input pointer.

2.5.8.2.104 #define CC_ECPKI_SM2_INVALID_EPHEMERAL_PUB_OUT_PTR (CC_ECPKI_MODULE_ERROR_BASE + 0xA8UL)

Illegal ephemeral public key output pointer.

2.5.8.2.105 #define CC_ECPKI_SM2_INVALID_ID_PTR (CC_ECPKI_MODULE_ERROR_BASE + 0xA1UL)

Illegal key ID pointer.

2.5.8.2.106 #define CC_ECPKI_SM2_INVALID_ID_SIZE (CC_ECPKI_MODULE_ERROR_BASE + 0xA2UL)

Illegal key ID size.

2.5.8.2.107 #define CC_ECPKI_SM2_INVALID_IN_PARAM_PTR (CC_ECPKI_MODULE_ERROR_BASE + 0xABUL)

Illegal key in parameter pointer.

2.5.8.2.108 #define CC_ECPKI_SM2_INVALID_IN_PARAM_SIZE (CC_ECPKI_MODULE_ERROR_BASE + 0xA3UL)

Illegal key in parameter size.

2.5.8.2.109 #define CC_ECPKI_SM2_INVALID_KE_CONTEXT_PTR (CC_ECPKI_MODULE_ERROR_BASE + 0xA0UL)

Illegal key context pointer.

2.5.8.2.110 #define CC_ECPKI_SM2_INVALID_OUT_PARAM_PTR (CC_ECPKI_MODULE_ERROR_BASE + 0xA5UL)

Illegal key out parameter pointer.

2.5.8.2.111 #define CC_ECPKI_SM2_INVALID_OUT_PARAM_SIZE (CC_ECPKI_MODULE_ERROR_BASE + 0xA4UL)

Illegal key out parameter size.

2.5.8.2.112 #define CC_ECPKI_SM2_INVALID_SHARED_SECRET_IN_PTR (CC_ECPKI_MODULE_ERROR_BASE + 0xAAUL)

Illegal shared secret input pointer.

2.5.8.2.113 #define CC_ECPKI_SM2_INVALID_SHARED_SECRET_OUT_PTR (CC_ECPKI_MODULE_ERROR_BASE + 0xA9UL)

Illegal shared secret output pointer.

2.5.9 CryptoCell ECPKI type definitions

Contains CryptoCell ECPKI type definitions.

2.5.9.1 Data structures

- struct **CCEcpkiPointAffine_t**
- struct **EcdsaSignContext_t**
- struct **CCEcdsaSignUserContext_t**

  The context definition of the user for the signing operation.
- struct **CCEcdsaFipsKatContext_t**
- struct **CCEcdhFipsKatContext_t**
- struct **CCEcpkiKgFipsContext_t**
- struct **CCEcpkiDomain_t**

  The structure containing the EC domain parameters in little-endian form.
- struct **CCEcpkiPublKey_t**
- struct **CCEcpkiUserPublKey_t**

  The user structure prototype of the EC public key.
- struct **CCEcpkiPrivKey_t**
- struct **CCEcpkiUserPrivKey_t**

  The user structure prototype of the EC private key.
- struct **CCEcdhTempData_t**
- struct **CCEcpkiBuildTempData_t**
- struct **EcdsaVerifyContext_t**
- struct **CCEcdsaVerifyUserContext_t**

  The context definition of the user for the verification operation.
- struct **CCEcpkiKgTempData_t**

- struct CCEciesTempData_t

2.5.9.2 Macros

- #define CC_ECPKI_FIPS_ORDER_LENGTH (256/CC_BITS_IN_BYTE)
- #define CC_PKA_DOMAIN_LLF_BUFF_SIZE_IN_WORDS (10 + 3*CC_ECPKI_MODUL_MAX_LENGTH_IN_WORDS)

2.5.9.3 typedefs

- typedef uint32_t CCEcdsaSignIntBuff_t[CC_PKA_ECDSA_SIGN_BUFF_MAX_LENGTH_IN_WORDS]
- typedef struct CCEcdsaSignUserContext_t CCEcdsaSignUserContext_t

  The context definition of the user for the signing operation.
- typedef struct CCEcdsaFipsKatContext_t CCEcdsaFipsKatContext_t
- typedef struct CCEcdhFipsKatContext_t CCEcdhFipsKatContext_t
- typedef struct CCEcpkiKgFipsContext_t CCEcpkiKgFipsContext_t
- typedef struct CCEcpkiUserPublKey_t CCEcpkiUserPublKey_t

  The user structure prototype of the EC public key.
- typedef struct CCEcpkiUserPrivKey_t CCEcpkiUserPrivKey_t

  The user structure prototype of the EC private key.
- typedef struct CCEcdhTempData_t CCEcdhTempData_t
- typedef struct CCEcpkiBuildTempData_t CCEcpkiBuildTempData_t
- typedef uint32_t CCEcdsaVerifyIntBuff_t[CC_PKA_ECDSA_VERIFY_BUFF_MAX_LENGTH_IN_WORDS]
- typedef struct CCEcdsaVerifyUserContext_t CCEcdsaVerifyUserContext_t

  The context definition of the user for the verification operation.
- typedef struct CCEcpkiKgTempData_t CCEcpkiKgTempData_t
- typedef struct CCEciesTempData_t CCEciesTempData_t

2.5.9.4 Enumerations

- enum CCEcpkiDomainID_t { CC_ECPKI_DomainID_secp192k1, CC_ECPKI_DomainID_secp192r1, CC_ECPKI_DomainID_secp224k1, CC_ECPKI_DomainID_secp224r1, CC_ECPKI_DomainID_secp256k1, CC_ECPKI_DomainID_secp256r1, CC_ECPKI_DomainID_secp384r1, CC_ECPKI_DomainID_secp521r1, CC_ECPKI_DomainID_bp256r1, CC_ECPKI_DomainID_Builded, CC_ECPKI_DomainID_sm2, CC_ECPKI_DomainID_OffMode, CC_ECPKI_DomainIDLast = 0x7FFFFFFF }

  EC domain identifiers.

- enum **CCEcpkiHashOpMode_t** { **CC_ECPKI_HASH_SHA1_mode** = 0, **CC_ECPKI_HASH_SHA224_mode** = 1, **CC_ECPKI_HASH_SHA256_mode** = 2, **CC_ECPKI_HASH_SHA384_mode** = 3, **CC_ECPKI_HASH_SHA512_mode** = 4, **CC_ECPKI_AFTER_HASH_SHA1_mode** = 5, **CC_ECPKI_AFTER_HASH_SHA224_mode** = 6, **CC_ECPKI_AFTER_HASH_SHA256_mode** = 7, **CC_ECPKI_AFTER_HASH_SHA384_mode** = 8, **CC_ECPKI_AFTER_HASH_SHA512_mode** = 9, **CC_ECPKI_HASH_NumOfModes**, **CC_ECPKI_HASH_OpModeLast** = 0x7FFFFFFF }

  Hash operation mode.

- enum **CCEcpkiPointCompression_t** { **CC_EC_PointCompressed** = 2, **CC_EC_PointUncompressed** = 4, **CC_EC_PointContWrong** = 5, **CC_EC_PointHybrid** = 6, **CC_EC_PointCompresOffMode** = 8, **CC_ECPKI_PointCompressionLast** = 0x7FFFFFFF }
- enum **ECPublKeyCheckMode_t** { **CheckPointersAndSizesOnly** = 0, **ECpublKeyPartlyCheck** = 1, **ECpublKeyFullCheck** = 2, **PublKeyChecingOffMode**, **EC_PublKeyCheckModeLast** = 0x7FFFFFFF }
- enum **CCEcpkiScaProtection_t** { **SCAP_Inactive**, **SCAP_Active**, **SCAP_OFF_MODE**, **SCAP_LAST** = 0x7FFFFFFF }

2.5.9.5 Macro definition documentation

2.5.9.5.1 #define CC_ECPKI_FIPS_ORDER_LENGTH (256/CC_BITS_IN_BYTE)

The order length for FIPS ECC tests.

2.5.9.5.2 #define CC_PKA_DOMAIN_LLF_BUFF_SIZE_IN_WORDS (10 + 3*CC_ECPKI_MODUL_MAX_LENGTH_IN_WORDS)

The size of the internal buffer in words.

2.5.9.6 typedef documentation

2.5.9.6.1 typedef struct CCEcdhFipsKatContext_t CCEcdhFipsKatContext_t

ECDH KAT data structures for FIPS certification.

2.5.9.6.2 typedef struct CCEcdhTempData_t CCEcdhTempData_t

The type of the ECDH temporary data.

2.5.9.6.3 typedef struct CCEcdsaFipsKatContext_t CCEcdsaFipsKatContext_t

ECDSA KAT data structures for FIPS certification. The ECDSA KAT tests are defined for domain 256r1.

2.5.9.6.4 typedef uint32_t CCEcdsaSignIntBuff_t[CC_PKA_ECDSA_SIGN_BUFF_MAX_LENGTH_IN_WORDS]

The internal buffer used in the signing process.

2.5.9.6.5 typedef struct CCEcdsaSignUserContext_t CCEcdsaSignUserContext_t

This context saves the state of the operation, and must be saved by the user until the end of the API flow.

2.5.9.6.6 typedef uint32_t CCEcdsaVerifyIntBuff_t[CC_PKA_ECDSA_VERIFY_BUFF_MAX_LENGTH_IN_WORDS]

The internal buffer used in the verification process.

2.5.9.6.7 typedef struct CCEcdsaVerifyUserContext_t CCEcdsaVerifyUserContext_t

The context saves the state of the operation, and must be saved by the user until the end of the API flow.

2.5.9.6.8 typedef struct CCEciesTempData_t CCEciesTempData_t

The temporary data definition of the ECIES.

2.5.9.6.9 typedef struct CCEcpkiBuildTempData_t CCEcpkiBuildTempData_t

EC build temporary data.

2.5.9.6.10 typedef struct CCEcpkiKgFipsContext_t CCEcpkiKgFipsContext_t

ECPKI data structures for FIPS certification.

2.5.9.6.11 typedef struct CCEcpkiKgTempData_t CCEcpkiKgTempData_t

The temporary data type of the ECPKI KG.

2.5.9.6.12 typedef struct CCEcpkiUserPrivKey_t CCEcpkiUserPrivKey_t

This structure must be saved by the user. It is used as input to ECC functions, for example, CC_EcdsaSign().

2.5.9.6.13 typedef struct CCEcpkiUserPublKey_t CCEcpkiUserPublKey_t

This structure must be saved by the user. It is used as input to ECC functions, for example, CC_EcdsaVerify().

2.5.9.7 Enumeration type documentation

2.5.9.7.1 enum CCEcpkiDomainID_t

For more information, see *Standards for Efficient Cryptography Group (SECG): SEC2 Recommended Elliptic Curve Domain Parameters, Version 1.0* .

Enumerator:

Enum	Description
CC_ECPKI_DomainID_secp192k1	EC secp192k1.
CC_ECPKI_DomainID_secp192r1	EC secp192r1.
CC_ECPKI_DomainID_secp224k1	EC secp224k1.
CC_ECPKI_DomainID_secp224r1	EC secp224r1.
CC_ECPKI_DomainID_secp256k1	EC secp256k1.
CC_ECPKI_DomainID_secp256r1	EC secp256r1.
CC_ECPKI_DomainID_secp384r1	EC secp384r1.
CC_ECPKI_DomainID_secp521r1	EC secp521r1.
CC_ECPKI_DomainID_bp256r1	EC bp256r1.
CC_ECPKI_DomainID_Builded	User given, not identified.
CC_ECPKI_DomainID_sm2	SM2 domain.
CC_ECPKI_DomainID_OffMode	Reserved.
CC_ECPKI_DomainIDLast	Reserved.

2.5.9.7.2 enum CCEcpkiHashOpMode_t

Defines hash modes according to *IEEE 1363-2000: IEEE Standard for Standard Specifications for Public-Key Cryptography* .

Enumerator:

Enum	Description
CC_ECPKI_HASH_SHA1_mode	The message data will be hashed with SHA-1.
CC_ECPKI_HASH_SHA224_mode	The message data will be hashed with SHA-224.
CC_ECPKI_HASH_SHA256_mode	The message data will be hashed with SHA-256.
CC_ECPKI_HASH_SHA384_mode	The message data will be hashed with SHA-384.
CC_ECPKI_HASH_SHA512_mode	The message data will be hashed with SHA-512.
CC_ECPKI_AFTER_HASH_SHA1_mode	The message data is a digest of SHA-1 and will not be hashed.
CC_ECPKI_AFTER_HASH_SHA224_mode	The message data is a digest of SHA-224 and will not be hashed.
CC_ECPKI_AFTER_HASH_SHA256_mode	The message data is a digest of SHA-256 and will not be hashed.
CC_ECPKI_AFTER_HASH_SHA384_mode	The message data is a digest of SHA-384 and will not be hashed.
CC_ECPKI_AFTER_HASH_SHA512_mode	The message data is a digest of SHA-512 and will not be hashed.
CC_ECPKI_HASH_NumOfModes	The maximal number of hash modes.
CC_ECPKI_HASH_OpModeLast	Reserved.

2.5.9.7.3 enum CCEcpkiPointCompression_t

EC point-compression identifiers.

Enumerator:

Enum	Description
`CC_EC_PointCompressed`	A compressed point.
`CC_EC_PointUncompressed`	An uncompressed point.
`CC_EC_PointContWrong`	An incorrect point-control value.
`CC_EC_PointHybrid`	A hybrid point.
`CC_EC_PointCompresOffMode`	Reserved.
`CC_ECPKI_PointCompressionLast`	Reserved.

2.5.9.7.4 enum CCEcpkiScaProtection_t

SW SCA protection type.

Enumerator:

Enum	Description
`SCAP_Active`	SCA protection inactive.
`SCAP_OFF_MODE`	SCA protection active.
`SCAP_LAST`	Reserved.

2.5.9.7.5 enum ECPublKeyCheckMode_t

EC key checks.

Enumerator:

Enum	Description
`CheckPointersAndSizesOnly`	Check only preliminary input parameters.
`ECpublKeyPartlyCheck`	Check preliminary input parameters and verify that the EC public-key point is on the curve.
`ECpublKeyFullCheck`	Check preliminary input parameters, verify that the EC public-key point is on the curve, and verify that EC_GeneratorOrder*PubKey = 0
`EC_PublKeyCheckModeLast`	Reserved.

2.5.10 CryptoCell PAL APIs

Contains all PAL APIs and definitions.

2.5.10.1 Modules

- **CERT definitions**

Contains definitions that are used by the CERT related APIs.

- *The implementation of these functions need to be replaced according to the Platform and TEE_OS.* **CryptoCell PAL DMA related APIs**

  Contains definitions that are used for DMA-related APIs.

- **CryptoCell PAL TRNG APIs**

  Contains APIs for retrieving TRNG user parameters.

- **CryptoCell PAL abort operations**

  Contains CryptoCell PAL abort operations.

- **CryptoCell PAL entry or exit point APIs**

  Contains PAL initialization and termination APIs.

- **CryptoCell PAL logging APIs and definitions**

  Contains CryptoCell PAL layer log definitions.

- **CryptoCell PAL memory Barrier APIs**

  Contains memory-barrier implementation definitions and APIs.

- **CryptoCell PAL memory mapping APIs**

  Contains memory mapping functions.

- **CryptoCell PAL memory operations**

  Contains memory-operation functions.

- **CryptoCell PAL mutex APIs**

  Contains resource management functions.

- **CryptoCell PAL platform-dependent compiler-specific definitions**

  Contains CryptoCell PAL platform-dependent compiler-related definitions.

- **CryptoCell PAL power-management APIs**

  Contains PAL power-management APIs.

- **CryptoCell platform-dependent PAL layer definitions and types**

  Contains platform-dependent definitions and types of the PAL layer.

- **Specific errors of the CryptoCell PAL APIs**

Contains platform-dependent PAL-API error definitions.

2.5.11 CryptoCell PAL DMA related APIs

Contains definitions that are used for DMA-related APIs.

2.5.11.1 Data structures

- struct **CCPalDmaBlockInfo_t**

  User buffer scatter information.

2.5.11.2 Macros

- #define **SET_WORD_LE**

2.5.11.3 typedefs

- typedef void ***CC_PalDmaBufferHandle**

2.5.11.4 Enumerations

- enum **CCPalDmaBufferDirection_t** { **CC_PAL_DMA_DIR_NONE** = 0, **CC_PAL_DMA_DIR_TO_DEVICE** = 1, **CC_PAL_DMA_DIR_FROM_DEVICE** = 2, **CC_PAL_DMA_DIR_BI_DIRECTION** = 3, **CC_PAL_DMA_DIR_MAX**, **CC_PAL_DMA_DIR_RESERVE32** = 0x7FFFFFFF }

2.5.11.5 Functions

- uint32_t **CC_PalDmaBufferMap** (uint8_t *pDataBuffer, uint32_t buffSize, **CCPalDmaBufferDirection_t** copyDirection, uint32_t *pNumOfBlocks, **CCPalDmaBlockInfo_t** *pDmaBlockList, **CC_PalDmaBufferHandle** *dmaBuffHandle)

  This function is called by the CryptoCell runtime library before the HW is used. It maps a given data buffer (virtual address) for CryptoCell HW DMA use (physical address), and returns the list of one or more DMA-able (physical) blocks. Once it is called, only CryptoCell HW access to the buffer is allowed, until it is unmapped.

- uint32_t **CC_PalDmaBufferUnmap** (uint8_t *pDataBuffer, uint32_t buffSize, **CCPalDmaBufferDirection_t** copyDirection, uint32_t numOfBlocks, **CCPalDmaBlockInfo_t** *pDmaBlockList, **CC_PalDmaBufferHandle** dmaBuffHandle)

  This function is called by the CryptoCell runtime library after the HW is used. It unmaps a given buffer and frees its associated resources, if needed. It may unlock the buffer and flush it for CPU use. Once it is called, CryptoCell HW does not require any further access to this buffer.

- uint32_t **CC_PalDmaContigBufferAllocate** (uint32_t buffSize, uint8_t **ppVirtBuffAddr)

  Allocates a DMA-contiguous buffer for CPU use, and returns its virtual address. Before passing the buffer to the CryptoCell HW, **CC_PalDmaBufferMap** should be called.

- uint32_t **CC_PalDmaContigBufferFree** (uint32_t buffSize, uint8_t *pVirtBuffAddr)

  Frees resources previously allocated by **CC_PalDmaContigBufferAllocate**.

- uint32_t **CC_PalIsDmaBufferContiguous** (uint8_t *pDataBuffer, uint32_t buffSize)

  Checks whether the buffer is guaranteed to be a single contiguous DMA block.

2.5.11.6 Macro definition documentation

2.5.11.6.1 #define SET_WORD_LE

Definition for big to little endian.

2.5.11.7 typedef documentation

2.5.11.7.1 typedef void*CC_PalDmaBufferHandle

Definition for DMA buffer handle.

2.5.11.8 Enumeration type documentation

2.5.11.8.1 enum CCPalDmaBufferDirection_t

DMA directions configuration.

Enumerator:

Enum	Description
CC_PAL_DMA_DIR_NONE	No direction.
CC_PAL_DMA_DIR_TO_DEVICE	The original buffer is the input to the operation. It should be copied or mapped to the temporary buffer prior to activating the HW on it.
CC_PAL_DMA_DIR_FROM_DEVICE	The temporary buffer holds the output of the HW. This API should copy or map it to the original output buffer.
CC_PAL_DMA_DIR_BI_DIRECTION	The result is written over the original data at the same address. Should be treated as CC_PAL_DMA_DIR_TO_DEVICE and CC_PAL_DMA_DIR_FROM_DEVICE .
CC_PAL_DMA_DIR_MAX	Maximal DMA direction options.
CC_PAL_DMA_DIR_RESERVE32	Reserved.

2.5.11.9 Function documentation

2.5.11.9.1 uint32_t CC_PalDmaBufferMap (uint8_t * *pDataBuffer*, uint32_t *buffSize*, CCPalDmaBufferDirection_t *copyDirection*, uint32_t * *pNumOfBlocks*, CCPalDmaBlockInfo_t * *pDmaBlockList*, CC_PalDmaBufferHandle * *dmaBuffHandle*)

If the data buffer was already mapped by the secure OS prior to calling the CryptoCell runtime library, this API does not have to perform any actual mapping operation, but only return the list of DMA-able blocks.

Returns:

A non-zero value in case of failure.

Parameters:

I/O	Parameter	Description
in	pDataBuffer	The address of the buffer to map.
in	buffSize	The buffer size in Bytes.
in	copyDirection	The copy direction of the buffer, according to **CCPalDmaBufferDirection_t**: • TO_DEVICE - the original buffer is the input to the operation, and this function should copy it to the temporary buffer, prior to the activating the HW on the temporary buffer. • FROM_DEVICE - not relevant for this API. • BI_DIRECTION - used when the cryptographic operation is "in-place", that is, the result of encryption or decryption is written over the original data at the same address. Should be treated by this API same as TO_DEVICE.
In,out	pNumOfBlocks	• In - The maximal number of blocks to fill. • Out - the actual number of blocks.
out	pDmaBlockList	The list of DMA-able blocks that the buffer maps to.
out	dmaBuffHandle	A handle to the private resources of the mapped buffer.

2.5.11.9.2 uint32_t CC_PalDmaBufferUnmap (uint8_t * *pDataBuffer*, uint32_t *buffSize*, CCPalDmaBufferDirection_t *copyDirection*, uint32_t *numOfBlocks*, CCPalDmaBlockInfo_t * *pDmaBlockList*, CC_PalDmaBufferHandle *dmaBuffHandle*)

If the data buffer was already unmapped by the secure OS prior to calling the CryptoCell runtime library, this API does not have to perform any unmapping operation, and the actual unmapping can be done by the secure OS outside the context of the CryptoCell runtime library.

Returns:

A non-zero value in case of failure.

Parameters:

I/O	Parameter	Description
in	pDataBuffer	The address of the buffer to unmap.
in	buffSize	The buffer size in Bytes.
in	copyDirection	The copy direction of the buffer, according to **CCPalDmaBufferDirection_t**: • TO_DEVICE - not relevant for this API. • FROM_DEVICE - the temporary buffer holds the output of the HW, and this API should copy it to the actual output buffer. • BI_DIRECTION - used when the cryptographic operation is "in-place", that is, the result of encryption or decryption is written over the original data at the same address. Should be treated by this API same as FROM_DEVICE.

I/O	Parameter	Description
in	`numOfBlocks`	The number of DMA-able blocks that the buffer maps to.
in	`pDmaBlockList`	The list of DMA-able blocks that the buffer maps to.
in	`dmaBuffHandle`	A handle to the private resources of the mapped buffer.

2.5.11.9.3 uint32_t CC_PalDmaContigBufferAllocate (uint32_t *buffSize*, uint8_t ** *ppVirtBuffAddr*)

The returned address must be aligned to 32bits.

Returns:

A non-zero value in case of failure.

Parameters:

I/O	Parameter	Description
in	`buffSize`	The buffer size in Bytes.
out	`ppVirtBuffAddr`	The virtual address of the allocated buffer.

2.5.11.9.4 uint32_t CC_PalDmaContigBufferFree (uint32_t *buffSize*, uint8_t * *pVirtBuffAddr*)

Returns:

A non-zero value in case of failure.

Parameters:

I/O	Parameter	Description
in	`buffSize`	The buffer size in Bytes.
in	`pVirtBuffAddr`	The virtual address of the buffer to free.

2.5.11.9.5 uint32_t CC_PalIsDmaBufferContiguous (uint8_t * *pDataBuffer*, uint32_t *buffSize*)

Returns:

TRUE if the buffer is guaranteed to be a single contiguous DMA block.

FALSE otherwise.

Parameters:

I/O	Parameter	Description
in	`pDataBuffer`	The address of the user buffer.
in	`buffSize`	The size of the user buffer.

2.5.12 CryptoCell PAL TRNG APIs

Contains APIs for retrieving TRNG user parameters.

2.5.12.1 Data structures

- struct **CC_PalTrngParams_t**

2.5.12.2 Functions

- **CCError_t CC_PalTrngParamGet** (**CC_PalTrngParams_t** *pTrngParams, size_t *pParamsSize)

  This function returns the TRNG user parameters.

2.5.12.3 Function documentation

2.5.12.3.1 CCError_t CC_PalTrngParamGet (CC_PalTrngParams_t * *pTrngParams*, size_t * *pParamsSize*)

Returns:

`0` on success.

A non-zero value on failure.

Parameters:

I/O	Parameter	Description
out	`pTrngParams`	A pointer to the TRNG user parameters.
in,out	`pParamsSize`	A pointer to the size of the TRNG-user-parameters structure used. Input: the function must verify its size is the same as **CC_PalTrngParams_t**. Output: the function returns the size of **CC_PalTrngParams_t** for library-size verification.

2.5.13 CryptoCell PAL abort operations

Contains CryptoCell PAL abort operations.

2.5.13.1 Functions

- void **CC_PalAbort** (const char *exp)

  This function performs the "Abort" operation. It must be implemented according to the speicific platform and OS.

2.5.14 CryptoCell PAL entry or exit point APIs

Contains PAL initialization and termination APIs.

2.5.14.1 Functions

- int **CC_PalInit** (void)

  This function performs all initializations that may be required by your PAL implementation, specifically by the DMA-able buffer scheme.

- void **CC_PalTerminate** (void)

  This function terminates the PAL implementation and frees the resources that were allocated by **CC_PalInit**.

2.5.14.2 Function documentation

2.5.14.2.1 int CC_PalInit (void)

The existing implementation allocates a contiguous memory pool that is later used by the CryptoCell implementation. If no initializations are needed in your environment, the function can be minimized to return OK. It is called by **CC_LibInit**.

Returns:

A non-zero value on failure.

2.5.14.2.2 void CC_PalTerminate (void)

Returns:

Void.

2.5.15 CryptoCell PAL logging APIs and definitions

Contains CryptoCell PAL layer log definitions.

2.5.15.1 Macros

- #define **CC_PAL_LOG_LEVEL_NULL** (-1)
- #define **CC_PAL_LOG_LEVEL_ERR** 0
- #define **CC_PAL_LOG_LEVEL_WARN** 1
- #define **CC_PAL_LOG_LEVEL_INFO** 2
- #define **CC_PAL_LOG_LEVEL_DEBUG** 3
- #define **CC_PAL_LOG_LEVEL_TRACE** 4
- #define **CC_PAL_LOG_LEVEL_DATA** 5
- #define **CC_PAL_LOG_CUR_COMPONENT** 0xFFFFFFFF
- #define **CC_PAL_LOG_CUR_COMPONENT_NAME** "CC"
- #define **CC_PAL_MAX_LOG_LEVEL CC_PAL_LOG_LEVEL_NULL**

- #define __CC_PAL_LOG_LEVEL_EVAL(level) level
- #define _CC_PAL_MAX_LOG_LEVEL __CC_PAL_LOG_LEVEL_EVAL(CC_PAL_MAX_LOG_LEVEL)
- #define _CC_PAL_LOG(level, format, ...)
- #define CC_PAL_LOG_ERR(...) do {} while (0)
- #define CC_PAL_LOG_WARN(...) do {} while (0)
- #define CC_PAL_LOG_INFO(...) do {} while (0)
- #define CC_PAL_LOG_DEBUG(...) do {} while (0)
- #define CC_PAL_LOG_DUMP_BUF(msg, buf, size) do {} while (0)
- #define CC_PAL_LOG_TRACE(...) do {} while (0)
- #define CC_PAL_LOG_DATA(...) do {} while (0)

2.5.15.2 Macro definition documentation

2.5.15.2.1 #define __CC_PAL_LOG_LEVEL_EVAL(level) level

Evaluate `CC_PAL_MAX_LOG_LEVEL` in case provided by caller.

2.5.15.2.2 #define _CC_PAL_LOG(level, format, ...)

```
if (CC_PAL_logMask & CC_PAL_LOG_CUR_COMPONENT) \
        CC_PalLog(CC_PAL_LOG_LEVEL_ ## level, "%s:%s:%d " format,
CC_PAL_LOG_CUR_COMPONENT_NAME, __func__,__LINE__, ##__VA_ARGS__)
```

Filter logging based on `logMask`, and dispatch to platform-specific logging mechanism.

2.5.15.2.3 #define _CC_PAL_MAX_LOG_LEVEL __CC_PAL_LOG_LEVEL_EVAL(CC_PAL_MAX_LOG_LEVEL)

The maximal log-level definition.

2.5.15.2.4 #define CC_PAL_LOG_CUR_COMPONENT 0xFFFFFFFF

Default log debugged component.

2.5.15.2.5 #define CC_PAL_LOG_CUR_COMPONENT_NAME "CC"

Default log debugged component.

2.5.15.2.6 #define CC_PAL_LOG_DATA(...) do {} while (0)

Log debug data.

2.5.15.2.7 #define CC_PAL_LOG_DEBUG(...) do {} while (0)

Log debug messages.

2.5.15.2.8 #define CC_PAL_LOG_DUMP_BUF(msg, buf, size) do {} while (0)

Log debug buffer.

2.5.15.2.9 #define CC_PAL_LOG_ERR(...) do {} while (0)

Log messages according to log level.

2.5.15.2.10 #define CC_PAL_LOG_INFO(...) do {} while (0)

Log messages according to log level.

2.5.15.2.11 #define CC_PAL_LOG_LEVEL_DATA 5

PAL log level - data.

2.5.15.2.12 #define CC_PAL_LOG_LEVEL_DEBUG 3

PAL log level - debug.

2.5.15.2.13 #define CC_PAL_LOG_LEVEL_ERR 0

PAL log level - error.

2.5.15.2.14 #define CC_PAL_LOG_LEVEL_INFO 2

PAL log level - info.

2.5.15.2.15 #define CC_PAL_LOG_LEVEL_NULL (-1)

PAL log level - disabled.

2.5.15.2.16 #define CC_PAL_LOG_LEVEL_TRACE 4

PAL log level - trace.

2.5.15.2.17 #define CC_PAL_LOG_LEVEL_WARN 1

PAL log level - warning.

2.5.15.2.18 #define CC_PAL_LOG_TRACE(...) do {} while (0)

Log debug trace.

2.5.15.2.19 #define CC_PAL_LOG_WARN(...) do {} while (0)

Log messages according to log level.

2.5.15.2.20 #define CC_PAL_MAX_LOG_LEVEL CC_PAL_LOG_LEVEL_NULL

Default debug log level, when debug is set to off.

2.5.16 CryptoCell PAL memory Barrier APIs

Contains memory-barrier implementation definitions and APIs.

2.5.16.1 Functions

- void **CC_PalWmb** (void)
- void **CC_PalRmb** (void)

2.5.16.2 Function documentation

2.5.16.2.1 void CC_PalRmb (void)

This macro puts the memory barrier before the read operation.

Returns:

None

2.5.16.2.2 void CC_PalWmb (void)

This macro puts the memory barrier after the write operation.

Returns:

None

2.5.17 CryptoCell PAL memory mapping APIs

Contains memory mapping functions.

2.5.17.1 Functions

- uint32_t **CC_PalMemMap** (CCDmaAddr_t physicalAddress, uint32_t mapSize, uint32_t **ppVirtBuffAddr)

  This function returns the base virtual address that maps the base physical address.

- uint32_t **CC_PalMemUnMap** (uint32_t *pVirtBuffAddr, uint32_t mapSize)

  This function unmaps a specified address range that was previously mapped by **CC_PalMemMap**.

2.5.17.2 Function documentation

2.5.17.2.1 uint32_t CC_PalMemMap (CCDmaAddr_t *physicalAddress*, uint32_t *mapSize*, uint32_t ** *ppVirtBuffAddr*)

Returns:

0 on success.

A non-zero value in case of failure.

Parameters:

I/O	Parameter	Description
in	`physicalAddress`	The starting physical address of the I/O range to be mapped.
in	`mapSize`	The number of Bytes that were mapped.
out	`ppVirtBuffAddr`	A pointer to the base virtual address to which the physical pages were mapped.

2.5.17.2.2 uint32_t CC_PalMemUnMap (uint32_t * *pVirtBuffAddr*, uint32_t *mapSize*)

Returns:

`0` on success.

A non-zero value in case of failure.

Parameters:

I/O	Parameter	Description
in	`pVirtBuffAddr`	A pointer to the base virtual address to which the physical pages were mapped.
in	`mapSize`	The number of Bytes that were mapped.

2.5.18 CryptoCell PAL memory operations

Contains memory-operation functions.

2.5.18.1 Macros

- #define **CC_PalMemCmp**(aTarget, aSource, aSize) CC_PalMemCmpPlat(aTarget, aSource, aSize)

  This function compares between two given buffers, according to the given size.

- #define **CC_PalMemCopy**(aDestination, aSource, aSize) CC_PalMemCopyPlat(aDestination, aSource, aSize)

  This function copies aSize Bytes from the source buffer to the destination buffer.

- #define **CC_PalMemMove**(aDestination, aSource, aSize) CC_PalMemMovePlat(aDestination, aSource, aSize)

  This function moves aSize Bytes from the source buffer to the destination buffer. This function supports overlapped buffers.

- #define **CC_PalMemSet**(aTarget, aChar, aSize) CC_PalMemSetPlat(aTarget, aChar, aSize)

  This function sets aSize Bytes of aChar in the given buffer.

- #define **CC_PalMemSetZero**(aTarget, aSize) CC_PalMemSetZeroPlat(aTarget, aSize)

This function sets aSize Bytes in the given buffer with zeroes.

- #define **CC_PalMemMalloc**(aSize) CC_PalMemMallocPlat(aSize)

  This function allocates a memory buffer according to aSize .

- #define **CC_PalMemRealloc**(aBuffer, aNewSize) CC_PalMemReallocPlat(aBuffer, aNewSize)

  This function reallocates a memory buffer according to aNewSize . The contents of the old buffer is moved to the new location.

- #define **CC_PalMemFree**(aBuffer) CC_PalMemFreePlat(aBuffer)

  This function frees a previously-allocated buffer.

2.5.18.2 Macro definition documentation

2.5.18.2.1 #define CC_PalMemCmp(aTarget, aSource, aSize) CC_PalMemCmpPlat(aTarget, aSource, aSize)

Parameters:

Parameter	Description
aSize	Size of buffer expressed in bytes.
aSource	Source buffer.
aTarget	Target buffer.

Returns:

The return values are according to operating-system return values.

2.5.18.2.2 #define CC_PalMemCopy(aDestination, aSource, aSize) CC_PalMemCopyPlat(aDestination, aSource, aSize)

Parameters:

Parameter	Description
aSize	Size of buffer expressed in bytes.
aSource	Source buffer.
aDestination	Destination buffer.

Returns:

Void.

2.5.18.2.3 #define CC_PalMemFree(aBuffer) CC_PalMemFreePlat(aBuffer)

Parameters:

Parameter	Description
aBuffer	Target buffer.

Returns:

Void.

2.5.18.2.4 #define CC_PalMemMalloc(aSize) CC_PalMemMallocPlat(aSize)

Parameters:

Parameter	Description
aSize	Size of buffer expressed in bytes.

Returns:

A pointer to the allocated buffer on success.

NULL on failure.

2.5.18.2.5 #define CC_PalMemMove(aDestination, aSource, aSize) CC_PalMemMovePlat(aDestination, aSource, aSize)

Parameters:

Parameter	Description
aSize	Size of buffer expressed in bytes.
aSource	Source buffer.
aDestination	Destination buffer.

Returns:

void.

2.5.18.2.6 #define CC_PalMemRealloc(aBuffer, aNewSize) CC_PalMemReallocPlat(aBuffer, aNewSize)

Parameters:

Parameter	Description
aBuffer	Target buffer.
aNewSize	New size of buffer expressed in bytes.

Returns:

A pointer to the newly-allocated buffer on success.

NULL on failure.

2.5.18.2.7 #define CC_PalMemSet(aTarget, aChar, aSize) CC_PalMemSetPlat(aTarget, aChar, aSize)

Parameters:

Parameter	Description
aSize	Size of buffer expressed in bytes.
aChar	Target character.
aTarget	Target buffer.

Returns:

Void.

2.5.18.2.8 #define CC_PalMemSetZero(aTarget, aSize) CC_PalMemSetZeroPlat(aTarget, aSize)

Parameters:

Parameter	Description
aSize	Size of buffer expressed in bytes.
aTarget	Target buffer

Returns:

Void.

2.5.19 CryptoCell PAL mutex APIs

Contains resource management functions.

2.5.19.1 Functions

- **CCError_t CC_PalMutexCreate** (CC_PalMutex *pMutexId)

  This function creates a mutex.

- **CCError_t CC_PalMutexDestroy** (CC_PalMutex *pMutexId)

  This function destroys a mutex.

- **CCError_t CC_PalMutexLock** (CC_PalMutex *pMutexId, uint32_t aTimeOut)

  This function waits for a mutex with aTimeOut . aTimeOut is specified in milliseconds. A value of aTimeOut=CC_INFINITE means that the function will not return.

- **CCError_t CC_PalMutexUnlock** (CC_PalMutex *pMutexId)

  This function releases the mutex.

2.5.19.2 Function documentation

2.5.19.2.1 CCError_t **CC_PalMutexCreate (CC_PalMutex * *pMutexId*)**

Returns:

0 on success.

A non-zero value on failure.

Parameters:

I/O	Parameter	Description
out	pMutexId	A pointer to the handle of the created mutex.

2.5.19.2.2 CCError_t **CC_PalMutexDestroy (CC_PalMutex * *pMutexId*)**

Returns:

0 on success.

A non-zero value on failure.

Parameters:

I/O	Parameter	Description
in	pMutexId	A pointer to handle of the mutex to destroy.

2.5.19.2.3 CCError_t **CC_PalMutexLock (CC_PalMutex * *pMutexId*, uint32_t *aTimeOut*)**

Returns:

0 on success.

A non-zero value on failure.

Parameters:

I/O	Parameter	Description
in	pMutexId	A pointer to handle of the mutex.
in	aTimeOut	The timeout in mSec, or CC_INFINITE.

2.5.19.2.4 CCError_t **CC_PalMutexUnlock (CC_PalMutex * *pMutexId*)**

Returns:

0 on success.

A non-zero value on failure.

Parameters:

I/O	Parameter	Description
in	pMutexId	A pointer to the handle of the mutex.

2.5.20 CryptoCell PAL platform-dependent compiler-specific definitions

Contains CryptoCell PAL platform-dependent compiler-related definitions.

2.5.20.1 Macros

- #define **CC_PAL_COMPILER_SECTION**(sectionName) __attribute__((section(sectionName)))
- #define **CC_PAL_COMPILER_KEEP_SYMBOL** __attribute__((used))
- #define **CC_PAL_COMPILER_ALIGN**(alignement) __attribute__((aligned(alignement)))
- #define **CC_PAL_COMPILER_FUNC_NEVER_RETURNS** __attribute__((noreturn))

- #define CC_PAL_COMPILER_FUNC_DONT_INLINE __attribute__((noinline))
- #define CC_PAL_COMPILER_TYPE_MAY_ALIAS __attribute__((__may_alias__))
- #define CC_PAL_COMPILER_SIZEOF_STRUCT_MEMBER(type_name, member_name) sizeof(((type_name *)0)->member_name)
- #define CC_ASSERT_CONCAT_(a, b) a##b
- #define CC_ASSERT_CONCAT(a, b) CC_ASSERT_CONCAT_(a, b)
- #define CC_PAL_COMPILER_ASSERT(cond, message) enum { CC_ASSERT_CONCAT(assert_line_, __LINE__) = 1/(!!(cond)) }

2.5.20.2 Macro definition documentation

2.5.20.2.1 #define CC_ASSERT_CONCAT(a, b) CC_ASSERT_CONCAT_(a, b)

Definition of assertion.

2.5.20.2.2 #define CC_ASSERT_CONCAT_(a, b) a##b

Definition of assertion.

2.5.20.2.3 #define CC_PAL_COMPILER_ALIGN(alignement) __attribute__((aligned(alignement)))

Make a given data item aligned (alignment in Bytes).

2.5.20.2.4 #define CC_PAL_COMPILER_ASSERT(cond, message) enum { CC_ASSERT_CONCAT(assert_line_, __LINE__) = 1/(!!(cond)) }

Definition of assertion.

2.5.20.2.5 #define CC_PAL_COMPILER_FUNC_DONT_INLINE __attribute__((noinline))

Prevent a function from being inlined.

2.5.20.2.6 #define CC_PAL_COMPILER_FUNC_NEVER_RETURNS __attribute__((noreturn))

Mark a function that never returns.

2.5.20.2.7 #define CC_PAL_COMPILER_KEEP_SYMBOL __attribute__((used))

Mark symbol as used, that is, prevent the garbage collector from dropping it.

2.5.20.2.8 #define CC_PAL_COMPILER_SECTION(sectionName) __attribute__((section(sectionName)))

Associate a symbol with a link section.

2.5.20.2.9 #define CC_PAL_COMPILER_SIZEOF_STRUCT_MEMBER(type_name, member_name) sizeof(((type_name *)0)->member_name)

Get the size of a structure-type member.

2.5.20.2.10 #define CC_PAL_COMPILER_TYPE_MAY_ALIAS __attribute__((__may_alias__))

Given data type might serve as an alias for another data-type pointer.

2.5.21 CryptoCell PAL power-management APIs

Contains PAL power-management APIs.

2.5.21.1 Functions

- void **CC_PalPowerDown** (void)

  This function powers down CryptoCell.

- void **CC_PalPowerUp** (void)

  This function powers up CryptoCell.

2.5.21.2 Function documentation

2.5.21.2.1 void CC_PalPowerDown (void)

Typically, it calls PMU to actually power down. When is returns, the CryptoCell is considered to be powered down and will not be accessed by the driver.

2.5.21.2.2 void CC_PalPowerUp (void)

Typically, it will call PMU to actually do power up. When is returns, the CryptoCell is guaranteed to be powered up and it is saved to be accessed by the driver.

2.5.22 CryptoCell SM3 specific errors

Contains the definitions of the CryptoCell SM3 errors.

2.5.22.1 Macros

- #define **CC_SM3_INVALID_USER_CONTEXT_POINTER_ERROR** (**CC_SM3_MODULE_ERROR_BASE** + 0x0UL)
- #define **CC_SM3_USER_CONTEXT_CORRUPTED_ERROR** (**CC_SM3_MODULE_ERROR_BASE** + 0x1UL)
- #define **CC_SM3_DATA_IN_POINTER_INVALID_ERROR** (**CC_SM3_MODULE_ERROR_BASE** + 0x2UL)

- #define **CC_SM3_DATA_SIZE_ILLEGAL** (**CC_SM3_MODULE_ERROR_BASE** + 0x3UL)
- #define **CC_SM3_INVALID_RESULT_BUFFER_POINTER_ERROR** (**CC_SM3_MODULE_ERROR_BASE** + 0x4UL)
- #define **CC_SM3_LAST_BLOCK_ALREADY_PROCESSED_ERROR** (**CC_SM3_MODULE_ERROR_BASE** + 0x5UL)
- #define **CC_SM3_ILLEGAL_PARAMS_ERROR** (**CC_SM3_MODULE_ERROR_BASE** + 0x6UL)
- #define **CC_SM3_CTX_SIZES_ERROR** (**CC_SM3_MODULE_ERROR_BASE** + 0x7UL)
- #define **CC_SM3_IS_NOT_SUPPORTED** (**CC_SM3_MODULE_ERROR_BASE** + 0x8UL)

2.5.22.2 Macro definition documentation

2.5.22.2.1 #define CC_SM3_CTX_SIZES_ERROR (CC_SM3_MODULE_ERROR_BASE + 0x7UL)

Illegal context size.

2.5.22.2.2 #define CC_SM3_DATA_IN_POINTER_INVALID_ERROR (CC_SM3_MODULE_ERROR_BASE + 0x2UL)

Illegal data in pointer.

2.5.22.2.3 #define CC_SM3_DATA_SIZE_ILLEGAL (CC_SM3_MODULE_ERROR_BASE + 0x3UL)

Illegal data in size.

2.5.22.2.4 #define CC_SM3_ILLEGAL_PARAMS_ERROR (CC_SM3_MODULE_ERROR_BASE + 0x6UL)

Illegal parameter.

2.5.22.2.5 #define CC_SM3_INVALID_RESULT_BUFFER_POINTER_ERROR (CC_SM3_MODULE_ERROR_BASE + 0x4UL)

Illegal result buffer pointer.

2.5.22.2.6 #define CC_SM3_INVALID_USER_CONTEXT_POINTER_ERROR (CC_SM3_MODULE_ERROR_BASE + 0x0UL)

SM3 module on the CryptoCell layer base address - 0x00F03000

Illegal context pointer.

2.5.22.2.7 #define CC_SM3_IS_NOT_SUPPORTED (CC_SM3_MODULE_ERROR_BASE + 0x8UL)

SM3 is not supported.

2.5.22.2.8 #define CC_SM3_LAST_BLOCK_ALREADY_PROCESSED_ERROR (CC_SM3_MODULE_ERROR_BASE + 0x5UL)

Last block was already processed (may happen if previous block was not a multiple of block size).

2.5.22.2.9 #define CC_SM3_USER_CONTEXT_CORRUPTED_ERROR (CC_SM3_MODULE_ERROR_BASE + 0x1UL)

Context is corrupted.

2.5.23 CryptoCell SM3 type definitions

Contains CryptoCell SM3 type definitions.

2.5.23.1 Data structures

- struct **CCSm3UserContext_t**

2.5.23.2 Macros

- #define **CC_SM3_RESULT_SIZE_IN_BITS** 256
- #define **CC_SM3_RESULT_SIZE_IN_BYTES** (**CC_SM3_RESULT_SIZE_IN_BITS** / **CC_BITS_IN_BYTE**)
- #define **CC_SM3_RESULT_SIZE_IN_WORDS** (CC_SM3_RESULT_SIZE_IN_BYTES / **CC_32BIT_WORD_SIZE**)
- #define **CC_SM3_BLOCK_SIZE_IN_BYTES** 64
- #define **CC_SM3_BLOCK_SIZE_IN_WORDS** 16
- #define **CC_SM3_UPDATE_DATA_MAX_SIZE_IN_BYTES** (1 << 61)
- #define **CC_SM3_USER_CTX_SIZE_IN_WORDS** 165

2.5.23.3 typedefs

- typedef uint8_t **CCSm3ResultBuf_t**[CC_SM3_RESULT_SIZE_IN_BYTES]
- typedef struct **CCSm3UserContext_t CCSm3UserContext_t**

2.5.23.4 Macro definition documentation

2.5.23.4.1 #define CC_SM3_BLOCK_SIZE_IN_BYTES 64

SM3 block size

2.5.23.4.2 #define CC_SM3_RESULT_SIZE_IN_BITS 256

The size of the SM3 result in words.

2.5.23.4.3 #define CC_SM3_UPDATE_DATA_MAX_SIZE_IN_BYTES (1 << 61)

The maximal data size for the update operation.

2.5.23.4.4 #define CC_SM3_USER_CTX_SIZE_IN_WORDS 165

The size of user context prototype (see **CCSm3UserContext_t**) in words.

2.5.23.5 typedef documentation

2.5.23.5.1 typedef uint8_t CCSm3ResultBuf_t[CC_SM3_RESULT_SIZE_IN_BYTES]

The SM3 result buffer.

2.5.23.5.2 typedef struct CCSm3UserContext_t CCSm3UserContext_t

The context prototype of the user. The argument type that is passed by the user to the SM3 APIs. The context saves the state of the operation, and must be saved by the user until the end of the API flow.

2.5.24 CryptoCell SM4 specific errors

Contains the definitions of the CryptoCell SM4 errors.

2.5.24.1 Macros

- #define **CC_SM4_INVALID_USER_CONTEXT_POINTER_ERROR** (**CC_SM4_MODULE_ERROR_BASE** + 0x00UL)
- #define **CC_SM4_INVALID_IV_POINTER_ERROR** (**CC_SM4_MODULE_ERROR_BASE** + 0x01UL)
- #define **CC_SM4_ILLEGAL_OPERATION_MODE_ERROR** (**CC_SM4_MODULE_ERROR_BASE** + 0x02UL)
- #define **CC_SM4_ILLEGAL_KEY_SIZE_ERROR** (**CC_SM4_MODULE_ERROR_BASE** + 0x03UL)
- #define **CC_SM4_INVALID_KEY_POINTER_ERROR** (**CC_SM4_MODULE_ERROR_BASE** + 0x04UL)
- #define **CC_SM4_INVALID_ENCRYPT_MODE_ERROR** (**CC_SM4_MODULE_ERROR_BASE** + 0x05UL)
- #define **CC_SM4_USER_CONTEXT_CORRUPTED_ERROR** (**CC_SM4_MODULE_ERROR_BASE** + 0x06UL)
- #define **CC_SM4_DATA_IN_POINTER_INVALID_ERROR** (**CC_SM4_MODULE_ERROR_BASE** + 0x07UL)
- #define **CC_SM4_DATA_OUT_POINTER_INVALID_ERROR** (**CC_SM4_MODULE_ERROR_BASE** + 0x08UL)
- #define **CC_SM4_DATA_IN_SIZE_ILLEGAL** (**CC_SM4_MODULE_ERROR_BASE** + 0x09UL)

- #define **CC_SM4_ILLEGAL_PARAMS_ERROR** (**CC_SM4_MODULE_ERROR_BASE** + 0x0AUL)
- #define **CC_SM4_ILLEGAL_INPLACE_ERROR** (**CC_SM4_MODULE_ERROR_BASE** + 0x0BUL)
- #define **CC_SM4_IS_NOT_SUPPORTED** (**CC_SM4_MODULE_ERROR_BASE** + 0xFFUL)

2.5.24.2 Macro definition documentation

2.5.24.2.1 #define CC_SM4_DATA_IN_POINTER_INVALID_ERROR (CC_SM4_MODULE_ERROR_BASE + 0x07UL)

Illegal data in pointer.

2.5.24.2.2 #define CC_SM4_DATA_IN_SIZE_ILLEGAL (CC_SM4_MODULE_ERROR_BASE + 0x09UL)

Illegal data in size.

2.5.24.2.3 #define CC_SM4_DATA_OUT_POINTER_INVALID_ERROR (CC_SM4_MODULE_ERROR_BASE + 0x08UL)

Illegal data out pointer.

2.5.24.2.4 #define CC_SM4_ILLEGAL_INPLACE_ERROR (CC_SM4_MODULE_ERROR_BASE + 0x0BUL)

Illegal inplace operation.

2.5.24.2.5 #define CC_SM4_ILLEGAL_KEY_SIZE_ERROR (CC_SM4_MODULE_ERROR_BASE + 0x03UL)

Illegal key size.

2.5.24.2.6 #define CC_SM4_ILLEGAL_OPERATION_MODE_ERROR (CC_SM4_MODULE_ERROR_BASE + 0x02UL)

Illegal operation.

2.5.24.2.7 #define CC_SM4_ILLEGAL_PARAMS_ERROR (CC_SM4_MODULE_ERROR_BASE + 0x0AUL)

Illegal parameters.

2.5.24.2.8 #define CC_SM4_INVALID_ENCRYPT_MODE_ERROR (CC_SM4_MODULE_ERROR_BASE + 0x05UL)

Illegal operation.

2.5.24.2.9 #define CC_SM4_INVALID_IV_POINTER_ERROR (CC_SM4_MODULE_ERROR_BASE + 0x01UL)

Illegal IV pointer.

2.5.24.2.10 #define CC_SM4_INVALID_KEY_POINTER_ERROR (CC_SM4_MODULE_ERROR_BASE + 0x04UL)

Illegal key pointer.

2.5.24.2.11 #define CC_SM4_INVALID_USER_CONTEXT_POINTER_ERROR (CC_SM4_MODULE_ERROR_BASE + 0x00UL)

CC_SM4_MODULE_ERROR_BASE - 0x00F03100

Illegal user context.

2.5.24.2.12 #define CC_SM4_IS_NOT_SUPPORTED (CC_SM4_MODULE_ERROR_BASE + 0xFFUL)

SM4 is not supported.

2.5.24.2.13 #define CC_SM4_USER_CONTEXT_CORRUPTED_ERROR (CC_SM4_MODULE_ERROR_BASE + 0x06UL)

User context corrupted.

2.5.25 CryptoCell SM4 type definitions

Contains CryptoCell SM4 type definitions.

2.5.25.1 Data structures

- struct **CCSm4UserContext_t**

2.5.25.2 Macros

- #define **CC_SM4_USER_CTX_SIZE_IN_WORDS** 131
- #define **CC_SM4_CRYPTO_BLOCK_SIZE_IN_WORDS** 4
- #define **CC_SM4_BLOCK_SIZE_IN_BYTES** (**CC_SM4_CRYPTO_BLOCK_SIZE_IN_WORDS** *sizeof(uint32_t))
- #define **CC_SM4_KEY_SIZE_IN_WORDS CC_SM4_CRYPTO_BLOCK_SIZE_IN_WORDS**
- #define **CC_SM4_KEY_SIZE_IN_BYTES** (**CC_SM4_KEY_SIZE_IN_WORDS** *sizeof(uint32_t))
- #define **CC_SM4_IV_SIZE_IN_WORDS CC_SM4_CRYPTO_BLOCK_SIZE_IN_WORDS**

- #define CC_SM4_IV_SIZE_IN_BYTES (CC_SM4_IV_SIZE_IN_WORDS *sizeof(uint32_t))

2.5.25.3 typedefs

- typedef uint8_t CCSm4Iv_t[CC_SM4_IV_SIZE_IN_BYTES]
- typedef uint8_t CCSm4Key_t[CC_SM4_KEY_SIZE_IN_BYTES]
- typedef struct CCSm4UserContext_t CCSm4UserContext_t

2.5.25.4 Enumerations

- enum CCSm4EncryptMode_t { CC_SM4_ENCRYPT = 0, CC_SM4_DECRYPT = 1, CC_SM4_NUM_OF_ENCRYPT_MODES, CC_SM4_ENCRYPT_MODE_LAST = 0x7FFFFFFF }
- enum CCSm4OperationMode_t { CC_SM4_MODE_ECB = 0, CC_SM4_MODE_CBC = 1, CC_SM4_MODE_CTR = 2, CC_SM4_MODE_OFB = 3, CC_SM4_NUM_OF_OPERATION_MODES, CC_SM4_OPERATION_MODE_LAST = 0x7FFFFFFF }

2.5.25.5 Macro definition documentation

2.5.25.5.1 #define CC_SM4_BLOCK_SIZE_IN_BYTES (CC_SM4_CRYPTO_BLOCK_SIZE_IN_WORDS *sizeof(uint32_t))

The size of the SM4 block in bytes.

2.5.25.5.2 #define CC_SM4_CRYPTO_BLOCK_SIZE_IN_WORDS 4

The size of the SM4 block in words.

2.5.25.5.3 #define CC_SM4_IV_SIZE_IN_BYTES (CC_SM4_IV_SIZE_IN_WORDS *sizeof(uint32_t))

The size of the IV buffer in bytes.

2.5.25.5.4 #define CC_SM4_IV_SIZE_IN_WORDS CC_SM4_CRYPTO_BLOCK_SIZE_IN_WORDS

The size of the IV buffer in words.

2.5.25.5.5 #define CC_SM4_KEY_SIZE_IN_BYTES (CC_SM4_KEY_SIZE_IN_WORDS *sizeof(uint32_t))

The size of the Key buffer in bytes.

2.5.25.5.6 #define CC_SM4_KEY_SIZE_IN_WORDS CC_SM4_CRYPTO_BLOCK_SIZE_IN_WORDS

The size of the Key buffer in words.

2.5.25.5.7 #define CC_SM4_USER_CTX_SIZE_IN_WORDS 131

The size of the user's context prototype (see **CCSm4UserContext_t**) in words.

2.5.25.6 typedef documentation

2.5.25.6.1 typedef uint8_t CCSm4Iv_t[CC_SM4_IV_SIZE_IN_BYTES]

Defines the IV buffer. A 16-byte array.

2.5.25.6.2 typedef uint8_t CCSm4Key_t[CC_SM4_KEY_SIZE_IN_BYTES]

Defines the SM4 key data buffer.

2.5.25.6.3 typedef struct CCSm4UserContext_t CCSm4UserContext_t

The context prototype of the user.

The argument type that is passed by the user to the SM4 APIs. The context saves the state of the operation, and must be saved by the user till the end of the API flow.

2.5.25.7 Enumeration type documentation

2.5.25.7.1 enum CCSm4EncryptMode_t

The SM4 operation:

- Encrypt
- Decrypt

Enumerator:

Enum	Description
CC_SM4_ENCRYPT	An SM4 encrypt operation.
CC_SM4_DECRYPT	An SM4 decrypt operation.
CC_SM4_NUM_OF_ENCRYPT_MODES	The maximal number of operations.
CC_SM4_ENCRYPT_MODE_LAST	Reserved.

2.5.25.7.2 enum CCSm4OperationMode_t

The SM4 operation mode.

Enumerator:

Enum	Description
CC_SM4_MODE_ECB	ECB mode.
CC_SM4_MODE_CBC	CBC mode.
CC_SM4_MODE_CTR	CTR mode.
CC_SM4_MODE_OFB	OFB mode.

Enum	Description
`CC_SM4_NUM_OF_OPERATION_MODES`	The maximal number of SM4 modes.
`CC_SM4_OPERATION_MODE_LAST`	Reserved.

2.5.26 CryptoCell TRNG specific errors

Contains the definitions of the CryptoCell TRNG errors.

2.5.26.1 Macros

- #define **CC_TRNG_INVALID_PARAMS_ERROR** (**CC_TRNG_MODULE_ERROR_BASE** + 0x0UL)

2.5.26.2 Macro definition documentation

2.5.26.2.1 #define CC_TRNG_INVALID_PARAMS_ERROR (CC_TRNG_MODULE_ERROR_BASE + 0x0UL)

TRNG module on the CryptoCell layer base address - 0x00F02F00

Illegal input parameters.

2.5.27 CryptoCell definitions

Contains CryptoCell definitions.

2.5.27.1 Modules

- **CryptoCell AES type definitions**

  Contains CryptoCell AES type definitions.

- **CryptoCell general certification definitions**
- **CryptoCell hash type definitions**

  Contains CryptoCell hash type definitions.

- **CryptoCell library enums and definitions**
- *Contains all the enums and definitions that are used for the CryptoCell library initialization and terminate APIs, as well as the APIs themselves.*
- **CryptoCell register APIs**

  Contains macro definitions for accessing Arm CryptoCell registers.

- **General base error codes for CryptoCell**

  Contains general base-error codes for CryptoCell.

- **PKA enums and definitions**

  Contains all the enums and definitions that are used in the PKA related code.
- **Specific errors of the CryptoCell utility module APIs**

  Contains utility API error definitions.
- **bit-field operations macros**

  Contains bit-field operation macros.

2.5.28 CryptoCell general certification definitions

2.5.28.1 Data structures

- union **CCCertKatContext_t**

2.5.28.2 Macros

- #define **CCEcpkiKgCertContext_t CCSm2KeyGenCHCertContext_t**

2.5.28.3 Macro definition documentation

2.5.28.3.1 #define CCEcpkiKgCertContext_t CCSm2KeyGenCHCertContext_t

Definition for SM2 key generation certification context.

2.5.29 CryptoCell hash type definitions

Contains CryptoCell hash type definitions.

2.5.29.1 Data structures

- struct **CCHashUserContext_t**

2.5.29.2 Macros

- #define **CC_HASH_USER_CTX_SIZE_IN_WORDS** 197
- #define **CC_HASH_RESULT_SIZE_IN_WORDS** 16
- #define **CC_HASH_MD5_DIGEST_SIZE_IN_BYTES** 16
- #define **CC_HASH_MD5_DIGEST_SIZE_IN_WORDS** 4
- #define **CC_HASH_SHA1_DIGEST_SIZE_IN_BYTES** 20
- #define **CC_HASH_SHA1_DIGEST_SIZE_IN_WORDS** 5
- #define **CC_HASH_SHA224_DIGEST_SIZE_IN_WORDS** 7
- #define **CC_HASH_SHA256_DIGEST_SIZE_IN_WORDS** 8
- #define **CC_HASH_SHA384_DIGEST_SIZE_IN_WORDS** 12

- #define CC_HASH_SHA512_DIGEST_SIZE_IN_WORDS 16
- #define CC_HASH_SHA224_DIGEST_SIZE_IN_BYTES 28
- #define CC_HASH_SHA256_DIGEST_SIZE_IN_BYTES 32
- #define CC_HASH_SHA384_DIGEST_SIZE_IN_BYTES 48
- #define CC_HASH_SHA512_DIGEST_SIZE_IN_BYTES 64
- #define CC_HASH_BLOCK_SIZE_IN_WORDS 16
- #define CC_HASH_BLOCK_SIZE_IN_BYTES 64
- #define CC_HASH_SHA512_BLOCK_SIZE_IN_WORDS 32
- #define CC_HASH_SHA512_BLOCK_SIZE_IN_BYTES 128
- #define CC_HASH_UPDATE_DATA_MAX_SIZE_IN_BYTES (1 << 29)

2.5.29.3 typedefs

- typedef uint32_t CCHashResultBuf_t[CC_HASH_RESULT_SIZE_IN_WORDS]
- typedef struct CCHashUserContext_t CCHashUserContext_t

2.5.29.4 Enumerations

- enum CCHashOperationMode_t { CC_HASH_SHA1_mode = 0, CC_HASH_SHA224_mode = 1, CC_HASH_SHA256_mode = 2, CC_HASH_SHA384_mode = 3, CC_HASH_SHA512_mode = 4, CC_HASH_MD5_mode = 5, CC_HASH_NumOfModes, CC_HASH_OperationModeLast = 0x7FFFFFFF }

2.5.29.5 Macro definition documentation

2.5.29.5.1 #define CC_HASH_BLOCK_SIZE_IN_BYTES 64

The size of the SHA-1 hash block in bytes.

2.5.29.5.2 #define CC_HASH_BLOCK_SIZE_IN_WORDS 16

The size of the SHA-1 hash block in words.

2.5.29.5.3 #define CC_HASH_MD5_DIGEST_SIZE_IN_BYTES 16

The size of the MD5 digest result in bytes.

2.5.29.5.4 #define CC_HASH_MD5_DIGEST_SIZE_IN_WORDS 4

The size of the MD5 digest result in words.

2.5.29.5.5 #define CC_HASH_RESULT_SIZE_IN_WORDS 16

The size of the hash result in words. The maximal size for SHA-512 is 512 bits.

2.5.29.5.6 #define CC_HASH_SHA1_DIGEST_SIZE_IN_BYTES 20

The size of the SHA-1 digest result in bytes.

2.5.29.5.7 #define CC_HASH_SHA1_DIGEST_SIZE_IN_WORDS 5

The size of the SHA-1 digest result in words.

2.5.29.5.8 #define CC_HASH_SHA224_DIGEST_SIZE_IN_BYTES 28

The size of the SHA-256 digest result in bytes.

2.5.29.5.9 #define CC_HASH_SHA224_DIGEST_SIZE_IN_WORDS 7

The size of the SHA-224 digest result in words.

2.5.29.5.10 #define CC_HASH_SHA256_DIGEST_SIZE_IN_BYTES 32

The size of the SHA-256 digest result in bytes.

2.5.29.5.11 #define CC_HASH_SHA256_DIGEST_SIZE_IN_WORDS 8

The size of the SHA-256 digest result in words.

2.5.29.5.12 #define CC_HASH_SHA384_DIGEST_SIZE_IN_BYTES 48

The size of the SHA-384 digest result in bytes.

2.5.29.5.13 #define CC_HASH_SHA384_DIGEST_SIZE_IN_WORDS 12

The size of the SHA-384 digest result in words.

2.5.29.5.14 #define CC_HASH_SHA512_BLOCK_SIZE_IN_BYTES 128

The size of the SHA-2 hash block in bytes.

2.5.29.5.15 #define CC_HASH_SHA512_BLOCK_SIZE_IN_WORDS 32

The size of the SHA-2 hash block in words.

2.5.29.5.16 #define CC_HASH_SHA512_DIGEST_SIZE_IN_BYTES 64

The size of the SHA-512 digest result in bytes.

2.5.29.5.17 #define CC_HASH_SHA512_DIGEST_SIZE_IN_WORDS 16

The size of the SHA-512 digest result in words.

2.5.29.5.18 #define CC_HASH_UPDATE_DATA_MAX_SIZE_IN_BYTES (1 << 29)

The maximal data size for the update operation.

2.5.29.5.19 #define CC_HASH_USER_CTX_SIZE_IN_WORDS 197

The size of user's context prototype (see **CCHashUserContext_t**) in words.

2.5.29.6 typedef documentation

2.5.29.6.1 typedef uint32_t CCHashResultBuf_t[CC_HASH_RESULT_SIZE_IN_WORDS]

The hash result buffer.

2.5.29.6.2 typedef struct CCHashUserContext_t CCHashUserContext_t

The context prototype of the user. The argument type that is passed by the user to the hash APIs. The context saves the state of the operation, and must be saved by the user until the end of the API flow.

2.5.29.7 Enumeration type documentation

2.5.29.7.1 enum CCHashOperationMode_t

The hash operation mode.

Enumerator:

Enum	Description
CC_HASH_SHA1_mode	SHA-1.
CC_HASH_SHA224_mode	SHA-224.
CC_HASH_SHA256_mode	SHA-256.
CC_HASH_SHA384_mode	SHA-384.
CC_HASH_SHA512_mode	SHA-512.
CC_HASH_MD5_mode	MD5.
CC_HASH_NumOfModes	The number of hash modes.
CC_HASH_OperationModeLast	Reserved.

2.5.30 CryptoCell library enums and definitions

Cntains all the enums and definitions that are used for the CryptoCell library initialization and terminate APIs, as well as the APIs themselves.

2.5.31 CryptoCell platform-dependent PAL layer definitions and types

Contains platform-dependent definitions and types of the PAL layer.

2.5.31.1 Macros

- #define **CC_SUCCESS** 0UL
- #define **CC_FAIL** 1UL
- #define **CC_OK** 0
- #define **CC_UNUSED_PARAM**(prm) ((void)prm)
- #define **CC_MAX_UINT32_VAL** (0xFFFFFFFF)
- #define **CC_MIN**(a, b) (((a) < (b)) ? (a): (b))
- #define **CC_MAX**(a, b) (((a) > (b)) ? (a): (b))
- #define **CALC_FULL_BYTES**(numBits) ((numBits)/**CC_BITS_IN_BYTE** + (((numBits) & (**CC_BITS_IN_BYTE**-1)) > 0))
- #define **CALC_FULL_32BIT_WORDS**(numBits) ((numBits)/**CC_BITS_IN_32BIT_WORD** + (((numBits) & (**CC_BITS_IN_32BIT_WORD**-1)) > 0))
- #define **CALC_32BIT_WORDS_FROM_BYTES**(sizeBytes) ((sizeBytes)/**CC_32BIT_WORD_SIZE** + (((sizeBytes) & (**CC_32BIT_WORD_SIZE**-1)) > 0))
- #define **CALC_32BIT_WORDS_FROM_64BIT_DWORD**(sizeWords) (sizeWords *__CC_32BIT_WORD_IN_64BIT_DWORD__)
- #define **ROUNDUP_BITS_TO_32BIT_WORD**(numBits) (**CALC_FULL_32BIT_WORDS**(numBits) ***CC_BITS_IN_32BIT_WORD**)
- #define **ROUNDUP_BITS_TO_BYTES**(numBits) (**CALC_FULL_BYTES**(numBits) ***CC_BITS_IN_BYTE**)
- #define **ROUNDUP_BYTES_TO_32BIT_WORD**(sizeBytes) (**CALC_32BIT_WORDS_FROM_BYTES**(sizeBytes) ***CC_32BIT_WORD_SIZE**)
- #define **CALC_WORDS_TO_BYTES**(numwords) ((numwords)***CC_32BIT_WORD_SIZE**)
- #define **CC_1K_SIZE_IN_BYTES** 1024
- #define **CC_BITS_IN_BYTE** 8
- #define **CC_BITS_IN_32BIT_WORD** 32
- #define **CC_32BIT_WORD_SIZE** 4
- #define **CC_32BIT_WORD_IN_64BIT_DWORD** 2

2.5.31.2 Enumerations

- enum **CCBool** { **CC_FALSE** = 0, **CC_TRUE** = 1 }

2.5.31.3 Macro definition documentation

2.5.31.3.1 #define CALC_32BIT_WORDS_FROM_64BIT_DWORD(sizeWords) (sizeWords *CC_32BIT_WORD_IN_64BIT_DWORD)

This macro calculates the number of full 32-bit words from 64-bits dwords.

2.5.31.3.2 #define CALC_32BIT_WORDS_FROM_BYTES(sizeBytes) ((sizeBytes)/CC_32BIT_WORD_SIZE + (((sizeBytes) & (CC_32BIT_WORD_SIZE-1)) > 0))

This macro calculates the number of full 32-bit words from bytes where three bytes are one word.

2.5.31.3.3 #define CALC_FULL_32BIT_WORDS(numBits) ((numBits)/CC_BITS_IN_32BIT_WORD + (((numBits) & (CC_BITS_IN_32BIT_WORD-1)) > 0))

This macro calculates the number of full 32-bit words from bits where 31 bits are one word.

2.5.31.3.4 #define CALC_FULL_BYTES(numBits) ((numBits)/CC_BITS_IN_BYTE + (((numBits) & (CC_BITS_IN_BYTE-1)) > 0))

This macro calculates the number of full bytes from bits, where seven bits are one byte.

2.5.31.3.5 #define CALC_WORDS_TO_BYTES(numwords) ((numwords)*CC_32BIT_WORD_SIZE)

This macro calculates the number of bytes from words.

2.5.31.3.6 #define CC_1K_SIZE_IN_BYTES 1024

Definition of 1 KB in bytes.

2.5.31.3.7 #define CC_32BIT_WORD_IN_64BIT_DWORD 2

Definition of number of 32-bits words in a 64-bits dword.

2.5.31.3.8 #define CC_32BIT_WORD_SIZE 4

Definition of number of bytes in a 32-bits word.

2.5.31.3.9 #define CC_BITS_IN_32BIT_WORD 32

Definition of number of bits in a 32-bits word.

2.5.31.3.10 #define CC_BITS_IN_BYTE 8

Definition of number of bits in a byte.

2.5.31.3.11 #define CC_FAIL 1UL

Failure.

2.5.31.3.12 #define CC_MAX(a, b) (((a) > (b)) ? (a): (b))

Definition for maximal calculation.

2.5.31.3.13 #define CC_MAX_UINT32_VAL (0xFFFFFFFF)

The maximal uint32 value.

2.5.31.3.14 #define CC_MIN(a, b) (((a) < (b)) ? (a): (b))

Definition for minimal calculation.

2.5.31.3.15 #define CC_OK 0

Success (OK).

2.5.31.3.16 #define CC_SUCCESS 0UL

Success.

2.5.31.3.17 #define CC_UNUSED_PARAM(prm) ((void)prm)

This macro handles unused parameters in the code, to avoid compilation warnings.

2.5.31.3.18 #define ROUNDUP_BITS_TO_32BIT_WORD(numBits) (CALC_FULL_32BIT_WORDS(numBits) *CC_BITS_IN_32BIT_WORD)

This macro rounds up bits to 32-bit words.

2.5.31.3.19 #define ROUNDUP_BITS_TO_BYTES(numBits) (CALC_FULL_BYTES(numBits) *CC_BITS_IN_BYTE)

This macro rounds up bits to bytes.

2.5.31.3.20 #define ROUNDUP_BYTES_TO_32BIT_WORD(sizeBytes) (CALC_32BIT_WORDS_FROM_BYTES(sizeBytes) *CC_32BIT_WORD_SIZE)

This macro rounds up bytes to 32-bit words.

2.5.31.4 Enumeration type documentation

2.5.31.4.1 enum CCBool

Definition of Boolean type.

Enumerator:

Enum	Description
CC_FALSE	Boolean false.
CC_TRUE	Boolean true.

2.5.32 CryptoCell random-number generation definitions.

Contains the CryptoCell random-number generation definitions.

2.5.32.1 Data structures

- struct **CCRndState_t**

  The structure for the RND state. This includes internal data that must be saved by the user between boots.

- struct **CCRndContext_t**

2.5.32.2 Macros

- #define **CC_RND_SEED_MAX_SIZE_WORDS** 12
- #define **CC_RND_ADDITINAL_INPUT_MAX_SIZE_WORDS** **CC_RND_SEED_MAX_SIZE_WORDS**
- #define **CC_RND_MAX_GEN_VECTOR_SIZE_BITS** 0x7FFFF
- #define **CC_RND_MAX_GEN_VECTOR_SIZE_BYTES** 0xFFFF
- #define **CC_RND_REQUESTED_SIZE_COUNTER** 0x3FFFF

2.5.32.3 typedefs

- typedef int(***CCRndGenerateVectWorkFunc_t**) (void *rndState_ptr, unsigned char *out_ptr, size_t outSizeBytes)

2.5.32.4 Functions

- **CIMPORT_C CCError_t CC_RndGenerateVectorInRange** (**CCRndGenerateVectWorkFunc_t** f_rng, void *p_rng, size_t rndSizeInBits, uint8_t *maxVect_ptr, uint8_t *rndVect_ptr)

  Generates a random vector with specific limitations by testing candidates (described and used in FIPS Publication 186-4: Digital Signature Standard (DSS): B.1.2, B.4.2 etc.).

2.5.32.5 Macro definition documentation

2.5.32.5.1 #define CC_RND_ADDITINAL_INPUT_MAX_SIZE_WORDS CC_RND_SEED_MAX_SIZE_WORDS

The maximal size of the additional input-data in words.

2.5.32.5.2 #define CC_RND_MAX_GEN_VECTOR_SIZE_BITS 0x7FFFF

The maximal size of the generated vector in bits.

2.5.32.5.3 #define CC_RND_MAX_GEN_VECTOR_SIZE_BYTES 0xFFFF

The maximal size of the generated random vector in bytes.

2.5.32.5.4 #define CC_RND_REQUESTED_SIZE_COUNTER 0x3FFFF

The maximal size of the generated vector in bytes.

2.5.32.5.5 #define CC_RND_SEED_MAX_SIZE_WORDS 12

The maximal size of the random seed in words.

2.5.32.6 typedef documentation

2.5.32.6.1 typedef int(*CCRndGenerateVectWorkFunc_t) (void *rndState_ptr, unsigned char *out_ptr, size_t outSizeBytes)

The RND vector-generation function pointer.

2.5.32.7 Function documentation

2.5.32.7.1 CIMPORT_C CCError_t **CC_RndGenerateVectorInRange** (CCRndGenerateVectWorkFunc_t ***f_rng*, void * *p_rng*, size_t *rndSizeInBits*, uint8_t * *maxVect_ptr*, uint8_t * *rndVect_ptr*)**

This function draws a random vector, compare it to the range limits, and if within range - return it in rndVect_ptr. If outside the range, the function continues retrying until a conforming vector is found, or the maximal retries limit is exceeded. If maxVect_ptr is provided, rndSizeInBits specifies its size, and the output vector must conform to the range [1 < rndVect < maxVect_ptr]. If maxVect_ptr is NULL, rndSizeInBits specifies the exact required vector size, and the output vector must be the exact same bit size (with its most significant bit = 1).

The RND module must be instantiated prior to invocation of this API.

Returns:

CC_OK on success.

A non-zero value from cc_rnd_error.h on failure.

Parameters:

I/O	Parameter	Description
in	`f_rng`	Pointer to DRBG function
in,out	`p_rng`	Pointer to the random context - the input to f_rng.
in	`rndSizeInBits`	The size in bits of the random vector required. The allowed size in range 2 <= rndSizeInBits < 2^19-1, bits.

I/O	Parameter	Description
in	`maxVect_ptr`	Pointer to the vector defining the upper limit for the random vector output, Given as little-endian byte array. If not NULL, its actual size is treated as [(rndSizeInBits+7)/8] bytes.
in,out	`rndVect_ptr`	Pointer to the output buffer for the random vector. Must be at least [(rndSizeInBits+7)/8] bytes. Treated as little-endian byte array.

2.5.33 CryptoCell random-number specific errors

Contains the definitions of the CryptoCell RND errors.

2.5.33.1 Macros

- #define CC_RND_DATA_OUT_POINTER_INVALID_ERROR (CC_RND_MODULE_ERROR_BASE + 0x0UL)
- #define CC_RND_CAN_NOT_GENERATE_RAND_IN_RANGE (CC_RND_MODULE_ERROR_BASE + 0x1UL)
- #define CC_RND_CPRNG_TEST_FAIL_ERROR (CC_RND_MODULE_ERROR_BASE + 0x2UL)
- #define CC_RND_ADDITIONAL_INPUT_BUFFER_NULL (CC_RND_MODULE_ERROR_BASE + 0x3UL)
- #define CC_RND_ADDITIONAL_INPUT_SIZE_ERROR (CC_RND_MODULE_ERROR_BASE + 0x4UL)
- #define CC_RND_DATA_SIZE_OVERFLOW_ERROR (CC_RND_MODULE_ERROR_BASE + 0x5UL)
- #define CC_RND_VECTOR_SIZE_ERROR (CC_RND_MODULE_ERROR_BASE + 0x6UL)
- #define CC_RND_RESEED_COUNTER_OVERFLOW_ERROR (CC_RND_MODULE_ERROR_BASE + 0x7UL)
- #define CC_RND_INSTANTIATION_NOT_DONE_ERROR (CC_RND_MODULE_ERROR_BASE + 0x8UL)
- #define CC_RND_TRNG_LOSS_SAMPLES_ERROR (CC_RND_MODULE_ERROR_BASE + 0x9UL)
- #define CC_RND_TRNG_TIME_EXCEED_ERROR (CC_RND_MODULE_ERROR_BASE + 0xAUL)
- #define CC_RND_TRNG_LOSS_SAMPLES_AND_TIME_EXCEED_ERROR (CC_RND_MODULE_ERROR_BASE + 0xBUL)
- #define CC_RND_IS_KAT_MODE_ERROR (CC_RND_MODULE_ERROR_BASE + 0xCUL)
- #define CC_RND_OPERATION_IS_NOT_SUPPORTED_ERROR (CC_RND_MODULE_ERROR_BASE + 0xDUL)
- #define CC_RND_STATE_VALIDATION_TAG_ERROR (CC_RND_MODULE_ERROR_BASE + 0xEUL)

- #define **CC_RND_IS_NOT_SUPPORTED** (**CC_RND_MODULE_ERROR_BASE** + 0xFUL)
- #define **CC_RND_GEN_VECTOR_FUNC_ERROR** (**CC_RND_MODULE_ERROR_BASE** + 0x14UL)
- #define **CC_RND_WORK_BUFFER_PTR_INVALID_ERROR** (**CC_RND_MODULE_ERROR_BASE** + 0x20UL)
- #define **CC_RND_ILLEGAL_AES_KEY_SIZE_ERROR** (**CC_RND_MODULE_ERROR_BASE** + 0x21UL)
- #define **CC_RND_ILLEGAL_DATA_PTR_ERROR** (**CC_RND_MODULE_ERROR_BASE** + 0x22UL)
- #define **CC_RND_ILLEGAL_DATA_SIZE_ERROR** (**CC_RND_MODULE_ERROR_BASE** + 0x23UL)
- #define **CC_RND_ILLEGAL_PARAMETER_ERROR** (**CC_RND_MODULE_ERROR_BASE** + 0x24UL)
- #define **CC_RND_STATE_PTR_INVALID_ERROR** (**CC_RND_MODULE_ERROR_BASE** + 0x25UL)
- #define **CC_RND_TRNG_ERRORS_ERROR** (**CC_RND_MODULE_ERROR_BASE** + 0x26UL)
- #define **CC_RND_CONTEXT_PTR_INVALID_ERROR** (**CC_RND_MODULE_ERROR_BASE** + 0x27UL)
- #define **CC_RND_VECTOR_OUT_PTR_ERROR** (**CC_RND_MODULE_ERROR_BASE** + 0x30UL)
- #define **CC_RND_VECTOR_OUT_SIZE_ERROR** (**CC_RND_MODULE_ERROR_BASE** + 0x31UL)
- #define **CC_RND_MAX_VECTOR_IS_TOO_SMALL_ERROR** (**CC_RND_MODULE_ERROR_BASE** + 0x32UL)
- #define **CC_RND_KAT_DATA_PARAMS_ERROR** (**CC_RND_MODULE_ERROR_BASE** + 0x33UL)
- #define **CC_RND_TRNG_KAT_NOT_SUPPORTED_ERROR** (**CC_RND_MODULE_ERROR_BASE** + 0x34UL)
- #define **CC_RND_SRAM_NOT_SUPPORTED_ERROR** (**CC_RND_MODULE_ERROR_BASE** + 0x35UL)
- #define **CC_RND_AES_ERROR** (**CC_RND_MODULE_ERROR_BASE** + 0x36UL)
- #define **CC_RND_MODE_MISMATCH_ERROR** (**CC_RND_MODULE_ERROR_BASE** + 0x37UL)

2.5.33.2 Macro definition documentation

2.5.33.2.1 #define CC_RND_ADDITIONAL_INPUT_BUFFER_NULL (CC_RND_MODULE_ERROR_BASE + 0x3UL)

Illegal additional data buffer.

2.5.33.2.2 #define CC_RND_ADDITIONAL_INPUT_SIZE_ERROR (CC_RND_MODULE_ERROR_BASE + 0x4UL)

Illegal additional data size.

2.5.33.2.3 #define CC_RND_AES_ERROR (CC_RND_MODULE_ERROR_BASE + 0x36UL)

AES operation failure.

2.5.33.2.4 #define CC_RND_CAN_NOT_GENERATE_RAND_IN_RANGE (CC_RND_MODULE_ERROR_BASE + 0x1UL)

Random generation in range failed.

2.5.33.2.5 #define CC_RND_CONTEXT_PTR_INVALID_ERROR (CC_RND_MODULE_ERROR_BASE + 0x27UL)

Illegal context pointer.

2.5.33.2.6 #define CC_RND_CPRNG_TEST_FAIL_ERROR (CC_RND_MODULE_ERROR_BASE + 0x2UL)

CPRNGT test failed.

2.5.33.2.7 #define CC_RND_DATA_OUT_POINTER_INVALID_ERROR (CC_RND_MODULE_ERROR_BASE + 0x0UL)

RND module on the CryptoCell layer base address - 0x00F00C00

Illegal output pointer.

2.5.33.2.8 #define CC_RND_DATA_SIZE_OVERFLOW_ERROR (CC_RND_MODULE_ERROR_BASE + 0x5UL)

Data size overflow.

2.5.33.2.9 #define CC_RND_GEN_VECTOR_FUNC_ERROR (CC_RND_MODULE_ERROR_BASE + 0x14UL)

Illegal generate vector function pointer.

2.5.33.2.10 #define CC_RND_ILLEGAL_AES_KEY_SIZE_ERROR (CC_RND_MODULE_ERROR_BASE + 0x21UL)

Illegal AES key size.

2.5.33.2.11 #define CC_RND_ILLEGAL_DATA_PTR_ERROR (CC_RND_MODULE_ERROR_BASE + 0x22UL)

Illegal data pointer.

2.5.33.2.12 #define CC_RND_ILLEGAL_DATA_SIZE_ERROR (CC_RND_MODULE_ERROR_BASE + 0x23UL)

Illegal data size.

2.5.33.2.13 #define CC_RND_ILLEGAL_PARAMETER_ERROR (CC_RND_MODULE_ERROR_BASE + 0x24UL)

Illegal parameter.

2.5.33.2.14 #define CC_RND_INSTANTIATION_NOT_DONE_ERROR (CC_RND_MODULE_ERROR_BASE + 0x8UL)

Instantiation was not yet called.

2.5.33.2.15 #define CC_RND_IS_KAT_MODE_ERROR (CC_RND_MODULE_ERROR_BASE + 0xCUL)

RND is in Known Answer Test mode.

2.5.33.2.16 #define CC_RND_IS_NOT_SUPPORTED (CC_RND_MODULE_ERROR_BASE + 0xFUL)

RND is not supported.

2.5.33.2.17 #define CC_RND_KAT_DATA_PARAMS_ERROR (CC_RND_MODULE_ERROR_BASE + 0x33UL)

Illegal Known Answer Tests parameters.

2.5.33.2.18 #define CC_RND_MAX_VECTOR_IS_TOO_SMALL_ERROR (CC_RND_MODULE_ERROR_BASE + 0x32UL)

Maximal vector size is too small.

2.5.33.2.19 #define CC_RND_MODE_MISMATCH_ERROR (CC_RND_MODULE_ERROR_BASE + 0x37UL)

TRNG mode mismatch between PAL and library.

2.5.33.2.20 #define CC_RND_OPERATION_IS_NOT_SUPPORTED_ERROR (CC_RND_MODULE_ERROR_BASE + 0xDUL)

RND operation not supported.

2.5.33.2.21 #define CC_RND_RESEED_COUNTER_OVERFLOW_ERROR (CC_RND_MODULE_ERROR_BASE + 0x7UL)

Reseed counter overflow - in case this error was returned instantiation or reseeding operation must be called.

2.5.33.2.22 #define CC_RND_SRAM_NOT_SUPPORTED_ERROR (CC_RND_MODULE_ERROR_BASE + 0x35UL)

SRAM memory is not defined.

2.5.33.2.23 #define CC_RND_STATE_PTR_INVALID_ERROR (CC_RND_MODULE_ERROR_BASE + 0x25UL)

Illegal RND state pointer.

2.5.33.2.24 #define CC_RND_STATE_VALIDATION_TAG_ERROR (CC_RND_MODULE_ERROR_BASE + 0xEUL)

RND validity check failed.

2.5.33.2.25 #define CC_RND_TRNG_ERRORS_ERROR (CC_RND_MODULE_ERROR_BASE + 0x26UL)

TRNG errors.

2.5.33.2.26 #define CC_RND_TRNG_KAT_NOT_SUPPORTED_ERROR (CC_RND_MODULE_ERROR_BASE + 0x34UL)

TRNG Known Answer Test not supported.

2.5.33.2.27 #define CC_RND_TRNG_LOSS_SAMPLES_AND_TIME_EXCEED_ERROR (CC_RND_MODULE_ERROR_BASE + 0xBUL)

TRNG loss of samples and time exceeded limitations.

2.5.33.2.28 #define CC_RND_TRNG_LOSS_SAMPLES_ERROR (CC_RND_MODULE_ERROR_BASE + 0x9UL)

TRNG loss of samples.

2.5.33.2.29 #define CC_RND_TRNG_TIME_EXCEED_ERROR (CC_RND_MODULE_ERROR_BASE + 0xAUL)

TRNG Time exceeded limitations.

2.5.33.2.30 #define CC_RND_VECTOR_OUT_PTR_ERROR (CC_RND_MODULE_ERROR_BASE + 0x30UL)

Illegal output vector pointer.

2.5.33.2.31 #define CC_RND_VECTOR_OUT_SIZE_ERROR (CC_RND_MODULE_ERROR_BASE + 0x31UL)

Illegal output vector size.

2.5.33.2.32 #define CC_RND_VECTOR_SIZE_ERROR (CC_RND_MODULE_ERROR_BASE + 0x6UL)

Illegal vector size.

2.5.33.2.33 #define CC_RND_WORK_BUFFER_PTR_INVALID_ERROR (CC_RND_MODULE_ERROR_BASE + 0x20UL)

Illegal work buffer pointer.

2.5.34 CryptoCell register APIs

Contains macro definitions for accessing Arm CryptoCell's registers.

2.5.34.1 Macros

- #define **SB_REG_ADDR**(base, reg_name) (base + CC_REG_OFFSET(CRY_KERNEL, reg_name))
- #define **SB_REG_ADDR_UNIT**(base, reg_name, unit) (base + CC_REG_OFFSET(unit, reg_name))
- #define **CC_REG_OFFSET**(unit_name, reg_name) (CC_BASE_ ## unit_name + CC_ ## reg_name ## _REG_OFFSET)
- #define **CC_REG_BIT_SHIFT**(reg_name, field_name) (CC_ ## reg_name ## _ ## field_name ## _BIT_SHIFT)
- #define **CC_REG_BIT_MASK**(reg_name, field_name) (**BITMASK**(CC_ ## reg_name ## _ ## field_name ## _BIT_SIZE) << (CC_ ## reg_name ## _ ## field_name ## _BIT_SHIFT))
- #define **CC_REG_BIT_SIZE**(reg_name, field_name) (CC_ ## reg_name ## _ ## field_name ## _BIT_SIZE)
- #define **CC_REG_FLD_GET**(unit_name, reg_name, fld_name, reg_val)
- #define **CC_REG_FLD_GET2**(unit_name, reg_name, fld_name, reg_val)
- #define **CC_REG_FLD_SET**(unit_name, reg_name, fld_name, reg_shadow_var, new_fld_val)

2.5.34.2 Macro definition documentation

2.5.34.2.1 #define CC_REG_FLD_GET(unit_name, reg_name, fld_name, reg_val)

```
(CC_ ## reg_name ## _ ## fld_name ## _BIT_SIZE == 0x20 ?                    \
    reg_val :                        \
    BITFIELD_GET(reg_val, CC_ ## reg_name ## _ ## fld_name ## _BIT_SHIFT,
\
             CC_ ## reg_name ## _ ## fld_name ## _BIT_SIZE))
```

Bit fields get

2.5.34.2.2 #define CC_REG_FLD_GET2(unit_name, reg_name, fld_name, reg_val)

```
(CC_ ## reg_name ## _ ## fld_name ## _BIT_SIZE == 0x20 ?                    \
    reg_val :                        \
    BITFIELD_GET(reg_val, CC_ ## reg_name ## _ ## fld_name ## _BIT_SHIFT,
\
             CC_ ## reg_name ## _ ## fld_name ## _BIT_SIZE))
```

Bit fields access

2.5.34.2.3 #define CC_REG_FLD_SET(unit_name, reg_name, fld_name, reg_shadow_var, new_fld_val)

```
do {                                                                    \
    if (CC_ ## reg_name ## _ ## fld_name ## _BIT_SIZE == 0x20)          \
        reg_shadow_var = new_fld_val; \
  else                        \
   BITFIELD_SET(reg_shadow_var,                              \
            CC_ ## reg_name ## _ ## fld_name ## _BIT_SHIFT, \
            CC_ ## reg_name ## _ ## fld_name ## _BIT_SIZE,  \
            new_fld_val);                                   \
} while (0)
```

Bit fields set

2.5.35 CryptoCell true-random-number generation definitions.

Contains the CryptoCell true-random-number generation defines.

2.5.35.1 Data structures

- struct CCTrngWorkBuff_t
- struct CCTrngParams_t
- struct CCTrngState_t

2.5.35.2 Macros

- #define **CC_TRNG_WORK_BUFFER_SIZE_WORDS** 136
- #define **CC_RND_TRNG_SRC_INNER_OFFSET_WORDS** 2
- #define **CC_RND_TRNG_SRC_INNER_OFFSET_BYTES** (**CC_RND_TRNG_SRC_INNER_OFFSET_WORDS***sizeof(uint32_t))

2.5.35.3 typedefs

- typedef struct **CCTrngWorkBuff_t CCTrngWorkBuff_t**
- typedef struct **CCTrngParams_t CCTrngParams_t**
- typedef struct **CCTrngState_t CCTrngState_t**

2.5.35.4 Macro definition documentation

2.5.35.4.1 #define CC_RND_TRNG_SRC_INNER_OFFSET_BYTES (CC_RND_TRNG_SRC_INNER_OFFSET_WORDS *sizeof(uint32_t))

The definition of the internal offset in bytes.

2.5.35.4.2 #define CC_RND_TRNG_SRC_INNER_OFFSET_WORDS 2

The definition of the internal offset in words.

2.5.35.4.3 #define CC_TRNG_WORK_BUFFER_SIZE_WORDS 136

The size of the temporary buffer in words.

2.5.35.5 typedef documentation

2.5.35.5.1 typedef struct CCTrngParams_t CCTrngParams_t

The CC Random Generator Parameters structure **CCTrngParams_t** - containing the user given parameters and characterization values.

2.5.35.5.2 typedef struct CCTrngState_t CCTrngState_t

The structure for the RND state. This includes internal data that must be saved by the user between boots.

2.5.35.5.3 typedef struct CCTrngWorkBuff_t CCTrngWorkBuff_t

The definition of the RAM buffer, for internal use in instantiation or reseeding operations.

2.5.36 General base error codes for CryptoCell

Contains general base-error codes for CryptoCell.

2.5.36.1 Macros

- #define **CC_ERROR_BASE** 0x00F00000UL
- #define **CC_ERROR_LAYER_RANGE** 0x00010000UL
- #define **CC_ERROR_MODULE_RANGE** 0x00000100UL
- #define **CC_LAYER_ERROR_IDX** 0x00UL
- #define **LLF_LAYER_ERROR_IDX** 0x01UL
- #define **GENERIC_ERROR_IDX** 0x05UL
- #define **AES_ERROR_IDX** 0x00UL
- #define **DES_ERROR_IDX** 0x01UL
- #define **HASH_ERROR_IDX** 0x02UL
- #define **HMAC_ERROR_IDX** 0x03UL
- #define **RSA_ERROR_IDX** 0x04UL
- #define **DH_ERROR_IDX** 0x05UL
- #define **ECPKI_ERROR_IDX** 0x08UL
- #define **RND_ERROR_IDX** 0x0CUL
- #define **COMMON_ERROR_IDX** 0x0DUL
- #define **KDF_ERROR_IDX** 0x11UL
- #define **HKDF_ERROR_IDX** 0x12UL
- #define **AESCCM_ERROR_IDX** 0x15UL
- #define **FIPS_ERROR_IDX** 0x17UL
- #define **CH_CERT_ERROR_IDX** 0x18UL
- #define **PKA_MODULE_ERROR_IDX** 0x21UL
- #define **CHACHA_ERROR_IDX** 0x22UL
- #define **EC_MONT_EDW_ERROR_IDX** 0x23UL
- #define **CHACHA_POLY_ERROR_IDX** 0x24UL
- #define **POLY_ERROR_IDX** 0x25UL
- #define **SRP_ERROR_IDX** 0x26UL
- #define **AESGCM_ERROR_IDX** 0x27UL
- #define **AES_KEYWRAP_ERROR_IDX** 0x28UL
- #define **MNG_ERROR_IDX** 0x29UL
- #define **PROD_ERROR_IDX** 0x2AUL
- #define **FFCDH_ERROR_IDX** 0x2BUL
- #define **FFC_DOMAIN_ERROR_IDX** 0x2CUL

- #define SB_ECC_ERROR_IDX_ 0x2DUL
- #define EXT_DMA_ERROR_IDX 0x2EUL
- #define TRNG_ERROR_IDX 0x2FUL
- #define SM3_ERROR_IDX 0x30UL
- #define SM4_ERROR_IDX 0x31UL
- #define CPP_ERROR_IDX 0x32UL
- #define AXI_CTRL_ERROR_IDX 0x33UL
- #define CC_AES_MODULE_ERROR_BASE
- #define CC_DES_MODULE_ERROR_BASE
- #define CC_HASH_MODULE_ERROR_BASE
- #define CC_HMAC_MODULE_ERROR_BASE
- #define CC_RSA_MODULE_ERROR_BASE
- #define CC_DH_MODULE_ERROR_BASE
- #define CC_ECPKI_MODULE_ERROR_BASE
- #define LLF_ECPKI_MODULE_ERROR_BASE
- #define CC_RND_MODULE_ERROR_BASE
- #define LLF_RND_MODULE_ERROR_BASE
- #define CC_COMMON_MODULE_ERROR_BASE
- #define CC_KDF_MODULE_ERROR_BASE
- #define CC_HKDF_MODULE_ERROR_BASE
- #define CC_AESCCM_MODULE_ERROR_BASE
- #define CC_FIPS_MODULE_ERROR_BASE
- #define CC_CH_CERT_MODULE_ERROR_BASE
- #define PKA_MODULE_ERROR_BASE
- #define CC_CHACHA_MODULE_ERROR_BASE
- #define CC_EC_MONT_EDW_MODULE_ERROR_BASE
- #define CC_CHACHA_POLY_MODULE_ERROR_BASE
- #define CC_POLY_MODULE_ERROR_BASE
- #define CC_SRP_MODULE_ERROR_BASE
- #define CC_AESGCM_MODULE_ERROR_BASE
- #define CC_AES_KEYWRAP_MODULE_ERROR_BASE
- #define CC_MNG_MODULE_ERROR_BASE
- #define CC_PROD_MODULE_ERROR_BASE
- #define CC_FFCDH_MODULE_ERROR_BASE

- #define CC_FFC_DOMAIN_MODULE_ERROR_BASE
- #define CC_EXT_DMA_MODULE_ERROR_BASE
- #define CC_TRNG_MODULE_ERROR_BASE
- #define CC_SM3_MODULE_ERROR_BASE
- #define CC_SM4_MODULE_ERROR_BASE
- #define CC_CPP_MODULE_ERROR_BASE
- #define CC_AXI_CTRL_MODULE_ERROR_BASE
- #define GENERIC_ERROR_BASE (CC_ERROR_BASE + (CC_ERROR_LAYER_RANGE *GENERIC_ERROR_IDX))
- #define CC_FATAL_ERROR (GENERIC_ERROR_BASE + 0x00UL)
- #define CC_OUT_OF_RESOURCE_ERROR (GENERIC_ERROR_BASE + 0x01UL)
- #define CC_ILLEGAL_RESOURCE_VAL_ERROR (GENERIC_ERROR_BASE + 0x02UL)
- #define CC_CRYPTO_RETURN_ERROR(retCode, retcodeInfo, funcHandler) ((retCode) == 0 ? CC_OK: funcHandler(retCode, retcodeInfo))

2.5.36.2 Macro definition documentation

2.5.36.2.1 #define AES_ERROR_IDX 0x00UL

The AES error index.

2.5.36.2.2 #define AES_KEYWRAP_ERROR_IDX 0x28UL

The AES key-wrap error index.

2.5.36.2.3 #define AESCCM_ERROR_IDX 0x15UL

The AESCCM error index.

2.5.36.2.4 #define AESGCM_ERROR_IDX 0x27UL

The AESGCM error index.

2.5.36.2.5 #define AXI_CTRL_ERROR_IDX 0x33UL

The AXI CTRL error index.

2.5.36.2.6 #define CC_AES_KEYWRAP_MODULE_ERROR_BASE

```
(CC_ERROR_BASE + \
                                        (CC_ERROR_LAYER_RANGE *CC_LAYER_ERROR_IDX) + \
                                        (CC_ERROR_MODULE_RANGE *AES_KEYWRAP_ERROR_IDX))
```

The error base address of the AES key-wrap module - 0x00F02800.

2.5.36.2.7 #define CC_AES_MODULE_ERROR_BASE

```
(CC_ERROR_BASE + \
                                              (CC_ERROR_LAYER_RANGE  *CC_LAYER_ERROR_IDX)
+ \
                                              (CC_ERROR_MODULE_RANGE  *AES_ERROR_IDX))
```

The error base address of the AES module - 0x00F00000.

2.5.36.2.8 #define CC_AESCCM_MODULE_ERROR_BASE

```
(CC_ERROR_BASE + \
                                                (CC_ERROR_LAYER_RANGE
*CC_LAYER_ERROR_IDX) + \
                                                (CC_ERROR_MODULE_RANGE
*AESCCM_ERROR_IDX))
```

The error base address of the AESCCM module - 0x00F01500.

2.5.36.2.9 #define CC_AESGCM_MODULE_ERROR_BASE

```
(CC_ERROR_BASE + \
                                                (CC_ERROR_LAYER_RANGE
*CC_LAYER_ERROR_IDX) + \
                                                (CC_ERROR_MODULE_RANGE
*AESGCM_ERROR_IDX))
```

The error base address of the AESGCM module - 0x00F02700.

2.5.36.2.10 #define CC_AXI_CTRL_MODULE_ERROR_BASE

```
(CC_ERROR_BASE + \
                                                  (CC_ERROR_LAYER_RANGE
*CC_LAYER_ERROR_IDX) + \
                                                  (CC_ERROR_MODULE_RANGE
*AXI_CTRL_ERROR_IDX))
```

The error base address of the AXI_CTRL module - 0x00F03200.

2.5.36.2.11 #define CC_CH_CERT_MODULE_ERROR_BASE

```
(CC_ERROR_BASE + \
                                                   (CC_ERROR_LAYER_RANGE
*CC_LAYER_ERROR_IDX) + \
                                                   (CC_ERROR_MODULE_RANGE
*CH_CERT_ERROR_IDX))
```

The error base address of the Chinese Certification module - 0x00F01800.

2.5.36.2.12 #define CC_CHACHA_MODULE_ERROR_BASE

```
(CC_ERROR_BASE + \
                                                    (CC_ERROR_LAYER_RANGE
*CC_LAYER_ERROR_IDX) + \
                                                    (CC_ERROR_MODULE_RANGE
*CHACHA_ERROR_IDX) )
```

The error base address of the ChaCha module - 0x00F02200.

2.5.36.2.13 #define CC_CHACHA_POLY_MODULE_ERROR_BASE

```
(CC_ERROR_BASE + \
                                                    (CC_ERROR_LAYER_RANGE
*CC_LAYER_ERROR_IDX) + \
                                                    (CC_ERROR_MODULE_RANGE
*CHACHA_POLY_ERROR_IDX) )
```

The error base address of the Chacha-POLY module - 0x00F02400.

2.5.36.2.14 #define CC_COMMON_MODULE_ERROR_BASE

```
(CC_ERROR_BASE + \
                                              (CC_ERROR_LAYER_RANGE *CC_LAYER_ERROR_IDX)
+ \
                                              (CC_ERROR_MODULE_RANGE
*COMMON_ERROR_IDX) )
```

The error base address of the common module - 0x00F00D00.

2.5.36.2.15 #define CC_CPP_MODULE_ERROR_BASE

```
(CC_ERROR_BASE + \
                                                  (CC_ERROR_LAYER_RANGE
*CC_LAYER_ERROR_IDX) + \
                                                  (CC_ERROR_MODULE_RANGE
*CPP_ERROR_IDX) )
```

The error base address of the CPP module - 0x00F03200.

2.5.36.2.16 #define CC_CRYPTO_RETURN_ERROR(retCode, retcodeInfo, funcHandler) ((retCode) == 0 ? CC_OK: funcHandler(retCode, retcodeInfo))

A macro that defines the CryptoCell return value.

2.5.36.2.17 #define CC_DES_MODULE_ERROR_BASE

```
(CC_ERROR_BASE + \
                                              (CC_ERROR_LAYER_RANGE *CC_LAYER_ERROR_IDX)
+ \
                                              (CC_ERROR_MODULE_RANGE *DES_ERROR_IDX) )
```

The error base address of the DES module - 0x00F00100.

2.5.36.2.18 #define CC_DH_MODULE_ERROR_BASE

```
(CC_ERROR_BASE + \
                                            (CC_ERROR_LAYER_RANGE *CC_LAYER_ERROR_IDX) +
\
                                            (CC_ERROR_MODULE_RANGE *DH_ERROR_IDX) )
```

The error base address of the DH module - 0x00F00500.

2.5.36.2.19 #define CC_EC_MONT_EDW_MODULE_ERROR_BASE

```
(CC_ERROR_BASE + \
                                                    (CC_ERROR_LAYER_RANGE
*CC_LAYER_ERROR_IDX) + \
                                                    (CC_ERROR_MODULE_RANGE
*EC_MONT_EDW_ERROR_IDX))
```

The error base address of the EC MONT_EDW module - 0x00F02300.

2.5.36.2.20 #define CC_ECPKI_MODULE_ERROR_BASE

```
(CC_ERROR_BASE + \
                                             (CC_ERROR_LAYER_RANGE *CC_LAYER_ERROR_IDX)
+ \
                                             (CC_ERROR_MODULE_RANGE *ECPKI_ERROR_IDX))
```

The error base address of the ECPKI module - 0x00F00800.

2.5.36.2.21 #define CC_ERROR_BASE 0x00F00000UL

The definitions of the error number-space used for the different modules

The error base number for CryptoCell.

2.5.36.2.22 #define CC_ERROR_LAYER_RANGE 0x00010000UL

The error range number assigned for each layer.

2.5.36.2.23 #define CC_ERROR_MODULE_RANGE 0x00000100UL

The error range number assigned to each module on its specified layer.

2.5.36.2.24 #define CC_EXT_DMA_MODULE_ERROR_BASE

```
(CC_ERROR_BASE + \
                                                    (CC_ERROR_LAYER_RANGE
*CC_LAYER_ERROR_IDX) + \
                                                    (CC_ERROR_MODULE_RANGE
*EXT_DMA_ERROR_IDX))
```

The error base address of the External DMA module - 0x00F02B00.

2.5.36.2.25 #define CC_FATAL_ERROR (GENERIC_ERROR_BASE + 0x00UL)

CryptoCell fatal error.

2.5.36.2.26 #define CC_FFC_DOMAIN_MODULE_ERROR_BASE

```
(CC_ERROR_BASE + \
                                                    (CC_ERROR_LAYER_RANGE
*CC_LAYER_ERROR_IDX) + \
                                                  (CC_ERROR_MODULE_RANGE
*FFC_DOMAIN_ERROR_IDX))
```

The error base address of the FFCDH module - 0x00F02B00.

2.5.36.2.27 #define CC_FFCDH_MODULE_ERROR_BASE

```
(CC_ERROR_BASE + \
                                                  (CC_ERROR_LAYER_RANGE
*CC_LAYER_ERROR_IDX) + \
                                                  (CC_ERROR_MODULE_RANGE
*FFCDH_ERROR_IDX))
```

The error base address of the FFCDH module - 0x00F02B00.

2.5.36.2.28 #define CC_FIPS_MODULE_ERROR_BASE

```
(CC_ERROR_BASE + \
                                              (CC_ERROR_LAYER_RANGE
*CC_LAYER_ERROR_IDX) + \
                                              (CC_ERROR_MODULE_RANGE *FIPS_ERROR_IDX))
```

The error base address of the FIPS module - 0x00F01700.

2.5.36.2.29 #define CC_HASH_MODULE_ERROR_BASE

```
(CC_ERROR_BASE + \
                                            (CC_ERROR_LAYER_RANGE *CC_LAYER_ERROR_IDX)
+ \
                                            (CC_ERROR_MODULE_RANGE *HASH_ERROR_IDX))
```

The error base address of the hash module - 0x00F00200.

2.5.36.2.30 #define CC_HKDF_MODULE_ERROR_BASE

```
(CC_ERROR_BASE + \
                                         (CC_ERROR_LAYER_RANGE *CC_LAYER_ERROR_IDX) + \
                                         (CC_ERROR_MODULE_RANGE *HKDF_ERROR_IDX))
```

The error base address of the HKDF module - 0x00F01100.

2.5.36.2.31 #define CC_HMAC_MODULE_ERROR_BASE

```
(CC_ERROR_BASE + \
                                            (CC_ERROR_LAYER_RANGE *CC_LAYER_ERROR_IDX)
+ \
                                            (CC_ERROR_MODULE_RANGE *HMAC_ERROR_IDX))
```

The error base address of the HMAC module - 0x00F00300.

2.5.36.2.32 #define CC_ILLEGAL_RESOURCE_VAL_ERROR (GENERIC_ERROR_BASE + 0x02UL)

CryptoCell illegal resource value error.

2.5.36.2.33 #define CC_KDF_MODULE_ERROR_BASE

```
(CC_ERROR_BASE  +  \
                                            (CC_ERROR_LAYER_RANGE  *CC_LAYER_ERROR_IDX)  +  \
                                            (CC_ERROR_MODULE_RANGE  *KDF_ERROR_IDX))
```

The error base address of the KDF module - 0x00F01100.

2.5.36.2.34 #define CC_LAYER_ERROR_IDX 0x00UL

The CryptoCell error-layer index.

2.5.36.2.35 #define CC_MNG_MODULE_ERROR_BASE

```
(CC_ERROR_BASE  +  \
                               (CC_ERROR_LAYER_RANGE  *CC_LAYER_ERROR_IDX)  +  \
                               (CC_ERROR_MODULE_RANGE  *MNG_ERROR_IDX))
```

The error base address of the Management module - 0x00F02900.

2.5.36.2.36 #define CC_OUT_OF_RESOURCE_ERROR (GENERIC_ERROR_BASE + 0x01UL)

CryptoCell out of resources error.

2.5.36.2.37 #define CC_POLY_MODULE_ERROR_BASE

```
(CC_ERROR_BASE  +  \
                                                      (CC_ERROR_LAYER_RANGE
*CC_LAYER_ERROR_IDX)  +  \
                                                      (CC_ERROR_MODULE_RANGE
*POLY_ERROR_IDX))
```

The error base address of the POLY module - 0x00F02500.

2.5.36.2.38 #define CC_PROD_MODULE_ERROR_BASE

```
(CC_ERROR_BASE  +  \
                                               (CC_ERROR_LAYER_RANGE  *CC_LAYER_ERROR_IDX)
+  \
                                               (CC_ERROR_MODULE_RANGE  *PROD_ERROR_IDX))
```

The error base address of the production library - 0x00F02A00

2.5.36.2.39 #define CC_RND_MODULE_ERROR_BASE

```
(CC_ERROR_BASE  +  \
                                               (CC_ERROR_LAYER_RANGE  *CC_LAYER_ERROR_IDX)
+  \
                                               (CC_ERROR_MODULE_RANGE  *RND_ERROR_IDX))
```

The error base address of the RND module - 0x00F00C00.

2.5.36.2.40 #define CC_RSA_MODULE_ERROR_BASE

```
(CC_ERROR_BASE  +  \
                                        (CC_ERROR_LAYER_RANGE *CC_LAYER_ERROR_IDX) +
\
                                        (CC_ERROR_MODULE_RANGE *RSA_ERROR_IDX))
```

The error base address of the RSA module - 0x00F00400.

2.5.36.2.41 #define CC_SM3_MODULE_ERROR_BASE

```
(CC_ERROR_BASE + \
                                              (CC_ERROR_LAYER_RANGE
*CC_LAYER_ERROR_IDX) + \
                                              (CC_ERROR_MODULE_RANGE
*SM3_ERROR_IDX))
```

The error base address of the SM3 module - 0x00F03000.

2.5.36.2.42 #define CC_SM4_MODULE_ERROR_BASE

```
(CC_ERROR_BASE + \
                                            (CC_ERROR_LAYER_RANGE
*CC_LAYER_ERROR_IDX) + \
                                            (CC_ERROR_MODULE_RANGE
*SM4_ERROR_IDX))
```

The error base address of the SM4 module - 0x00F03100.

2.5.36.2.43 #define CC_SRP_MODULE_ERROR_BASE

```
(CC_ERROR_BASE + \
                                              (CC_ERROR_LAYER_RANGE
*CC_LAYER_ERROR_IDX) + \
                                              (CC_ERROR_MODULE_RANGE
*SRP_ERROR_IDX))
```

The error base address of the SRP module - 0x00F02600.

2.5.36.2.44 #define CC_TRNG_MODULE_ERROR_BASE

```
(CC_ERROR_BASE + \
                                          (CC_ERROR_LAYER_RANGE *LLF_LAYER_ERROR_IDX)
+ \
                                          (CC_ERROR_MODULE_RANGE *TRNG_ERROR_IDX))
```

The error base address of the low-level TRNG module - 0x00F02F00.

2.5.36.2.45 #define CH_CERT_ERROR_IDX 0x18UL

The Chinese Certification error index.

2.5.36.2.46 #define CHACHA_ERROR_IDX 0x22UL

The ChaCha error index.

2.5.36.2.47 #define CHACHA_POLY_ERROR_IDX 0x24UL

The ChaCha-POLY error index.

2.5.36.2.48 #define COMMON_ERROR_IDX 0x0DUL

The Common error index.

2.5.36.2.49 #define CPP_ERROR_IDX 0x32UL

The CPP error index.

2.5.36.2.50 #define DES_ERROR_IDX 0x01UL

The DES error index.

2.5.36.2.51 #define DH_ERROR_IDX 0x05UL

The DH error index.

2.5.36.2.52 #define EC_MONT_EDW_ERROR_IDX 0x23UL

The EC Montgomery and Edwards error index.

2.5.36.2.53 #define ECPKI_ERROR_IDX 0x08UL

The ECPKI error index.

2.5.36.2.54 #define EXT_DMA_ERROR_IDX 0x2EUL

External DMA error index.

2.5.36.2.55 #define FFC_DOMAIN_ERROR_IDX 0x2CUL

The FFC domain error index.

2.5.36.2.56 #define FFCDH_ERROR_IDX 0x2BUL

The FFCDH error index.

2.5.36.2.57 #define FIPS_ERROR_IDX 0x17UL

The FIPS error index.

2.5.36.2.58 #define GENERIC_ERROR_BASE (CC_ERROR_BASE + (CC_ERROR_LAYER_RANGE *GENERIC_ERROR_IDX))

The generic error base address of the user - 0x00F50000

2.5.36.2.59 #define GENERIC_ERROR_IDX 0x05UL

The generic error-layer index.

2.5.36.2.60 #define HASH_ERROR_IDX 0x02UL

The hash error index.

2.5.36.2.61 #define HKDF_ERROR_IDX 0x12UL

The HKDF error index.

2.5.36.2.62 #define HMAC_ERROR_IDX 0x03UL

The HMAC error index.

2.5.36.2.63 #define KDF_ERROR_IDX 0x11UL

The KDF error index.

2.5.36.2.64 #define LLF_ECPKI_MODULE_ERROR_BASE

```
(CC_ERROR_BASE + \
                                             (CC_ERROR_LAYER_RANGE *LLF_LAYER_ERROR_IDX)
+ \
                                             (CC_ERROR_MODULE_RANGE *ECPKI_ERROR_IDX))
```

The error base address of the low-level ECPKI module - 0x00F10800.

2.5.36.2.65 #define LLF_LAYER_ERROR_IDX 0x01UL

The error-layer index for low-level functions.

2.5.36.2.66 #define LLF_RND_MODULE_ERROR_BASE

```
(CC_ERROR_BASE + \
                                             (CC_ERROR_LAYER_RANGE *LLF_LAYER_ERROR_IDX)
+ \
                                             (CC_ERROR_MODULE_RANGE *RND_ERROR_IDX))
```

The error base address of the low-level RND module - 0x00F10C00.

2.5.36.2.67 #define MNG_ERROR_IDX 0x29UL

Management error index.

2.5.36.2.68 #define PKA_MODULE_ERROR_BASE

```
(CC_ERROR_BASE + \
                                                    (CC_ERROR_LAYER_RANGE
*CC_LAYER_ERROR_IDX) + \
                                                    (CC_ERROR_MODULE_RANGE
*PKA_MODULE_ERROR_IDX))
```

The error base address of the PKA module - 0x00F02100.

2.5.36.2.69 #define PKA_MODULE_ERROR_IDX 0x21UL

The PKA error index.

2.5.36.2.70 #define POLY_ERROR_IDX 0x25UL

The POLY error index.

2.5.36.2.71 #define PROD_ERROR_IDX 0x2AUL

Production error index.

2.5.36.2.72 #define RND_ERROR_IDX 0x0CUL

The RND error index.

2.5.36.2.73 #define RSA_ERROR_IDX 0x04UL

The RSA error index.

2.5.36.2.74 #define SB_ECC_ERROR_IDX_ 0x2DUL

Don't change! Error definition, reserved for Sec.Boot ECDSA

2.5.36.2.75 #define SM3_ERROR_IDX 0x30UL

The SM3 error index.

2.5.36.2.76 #define SM4_ERROR_IDX 0x31UL

The SM4 error index.

2.5.36.2.77 #define SRP_ERROR_IDX 0x26UL

The SRP error index.

2.5.36.2.78 #define TRNG_ERROR_IDX 0x2FUL

The TRNG error index.

2.5.37 PKA enums and definitions

Contains all the enums and definitions that are used in the PKA related code.

2.5.37.1 Macros

- #define CC_RSA_MAXIMUM_MOD_BUFFER_SIZE_IN_WORDS ((CC_RSA_MAX_VALID_KEY_SIZE_VALUE_IN_BITS + CC_PKA_WORD_SIZE_IN_BITS) / CC_BITS_IN_32BIT_WORD)
- #define CC_ECPKI_MODUL_MAX_LENGTH_IN_BITS 521
- #define CC_PKA_BARRETT_MOD_TAG_BUFF_SIZE_IN_WORDS 5
- #define CC_PKA_ECPKI_BARRETT_MOD_TAG_BUFF_SIZE_IN_WORDS CC_PKA_BARRETT_MOD_TAG_BUFF_SIZE_IN_WORDS
- #define CC_PKA_BARRETT_MOD_TAG_SIZE_IN_WORDS (((CC_PKA_WORD_SIZE_IN_BITS + PKA_EXTRA_BITS - 1) + (CC_BITS_IN_32BIT_WORD - 1)) / CC_BITS_IN_32BIT_WORD)
- #define CC_PKA_MAXIMUM_MOD_BUFFER_SIZE_IN_WORDS CC_RSA_MAXIMUM_MOD_BUFFER_SIZE_IN_WORDS
- #define CC_PKA_PUB_KEY_BUFF_SIZE_IN_WORDS (2*CC_PKA_BARRETT_MOD_TAG_BUFF_SIZE_IN_WORDS)
- #define CC_PKA_PRIV_KEY_BUFF_SIZE_IN_WORDS (2*CC_PKA_BARRETT_MOD_TAG_BUFF_SIZE_IN_WORDS)
- #define CC_PKA_KGDATA_BUFF_SIZE_IN_WORDS (3*CC_PKA_BARRETT_MOD_TAG_BUFF_SIZE_IN_WORDS + 3*CC_PKA_MAXIMUM_MOD_BUFFER_SIZE_IN_WORDS)
- #define CC_ECPKI_MODUL_MAX_LENGTH_IN_WORDS 18
- #define CC_ECPKI_ORDER_MAX_LENGTH_IN_WORDS (CC_ECPKI_MODUL_MAX_LENGTH_IN_WORDS + 1)
- #define CC_PKA_DOMAIN_BUFF_SIZE_IN_WORDS (2*CC_PKA_BARRETT_MOD_TAG_BUFF_SIZE_IN_WORDS)
- #define COUNT_NAF_WORDS_PER_KEY_WORD 8
- #define CC_PKA_ECDSA_NAF_BUFF_MAX_LENGTH_IN_WORDS (COUNT_NAF_WORDS_PER_KEY_WORD*CC_ECPKI_ORDER_MAX_LENGTH_IN_WORDS + 1)
- #define CC_PKA_ECPKI_SCALAR_MUL_BUFF_MAX_LENGTH_IN_WORDS (CC_PKA_ECDSA_NAF_BUFF_MAX_LENGTH_IN_WORDS + CC_ECPKI_MODUL_MAX_LENGTH_IN_WORDS + 2)
- #define CC_PKA_ECPKI_BUILD_TMP_BUFF_MAX_LENGTH_IN_WORDS (3*CC_ECPKI_MODUL_MAX_LENGTH_IN_WORDS+CC_PKA_ECPKI_SCALAR_MUL_BUFF_MAX_LENGTH_IN_WORDS)
- #define CC_PKA_ECDSA_SIGN_BUFF_MAX_LENGTH_IN_WORDS (6*CC_ECPKI_MODUL_MAX_LENGTH_IN_WORDS+CC_PKA_ECPKI_SCALAR_MUL_BUFF_MAX_LENGTH_IN_WORDS)
- #define CC_PKA_ECDH_BUFF_MAX_LENGTH_IN_WORDS (2*CC_ECPKI_ORDER_MAX_LENGTH_IN_WORDS + CC_PKA_ECPKI_SCALAR_MUL_BUFF_MAX_LENGTH_IN_WORDS)

- #define CC_PKA_KG_BUFF_MAX_LENGTH_IN_WORDS (2*CC_ECPKI_ORDER_MAX_LENGTH_IN_WORDS + CC_PKA_ECPKI_SCALAR_MUL_BUFF_MAX_LENGTH_IN_WORDS)
- #define CC_PKA_ECDSA_VERIFY_BUFF_MAX_LENGTH_IN_WORDS (3*CC_ECPKI_MODUL_MAX_LENGTH_IN_WORDS)
- #define CC_PKA_WORD_SIZE_IN_BITS 128
- #define CC_RSA_MAX_VALID_KEY_SIZE_VALUE_IN_BITS 4096
- #define CC_RSA_MAX_KEY_GENERATION_HW_SIZE_BITS 4096
- #define BSV_CERT_RSA_KEY_SIZE_IN_BITS 2048
- #define BSV_CERT_RSA_KEY_SIZE_IN_BYTES (BSV_CERT_RSA_KEY_SIZE_IN_BITS/CC_BITS_IN_BYTE)
- #define BSV_CERT_RSA_KEY_SIZE_IN_WORDS (BSV_CERT_RSA_KEY_SIZE_IN_BITS/CC_BITS_IN_32BIT_WORD)
- #define PKA_EXTRA_BITS 8
- #define PKA_MAX_COUNT_OF_PHYS_MEM_REGS 32
- #define RSA_PKA_BARRETT_MOD_TAG_BUFF_SIZE_IN_WORDS 5
- #define RSA_PKA_BARRETT_MOD_TAG_BUFF_SIZE_IN_BYTES (RSA_PKA_BARRETT_MOD_TAG_BUFF_SIZE_IN_WORDS*CC_32BIT_WORD_SIZE)

2.5.37.2 Macro definition documentation

2.5.37.2.1 #define BSV_CERT_RSA_KEY_SIZE_IN_BITS 2048

Secure boot/debug certificate RSA public modulus key size in bits.

2.5.37.2.2 #define BSV_CERT_RSA_KEY_SIZE_IN_BYTES (BSV_CERT_RSA_KEY_SIZE_IN_BITS/CC_BITS_IN_BYTE)

Secure boot/debug certificate RSA public modulus key size in bytes.

2.5.37.2.3 #define BSV_CERT_RSA_KEY_SIZE_IN_WORDS (BSV_CERT_RSA_KEY_SIZE_IN_BITS/CC_BITS_IN_32BIT_WORD)

Secure boot/debug certificate RSA public modulus key size in words.

2.5.37.2.4 #define CC_ECPKI_MODUL_MAX_LENGTH_IN_BITS 521

The maximal EC modulus size.

2.5.37.2.5 #define CC_ECPKI_MODUL_MAX_LENGTH_IN_WORDS 18

The maximal size of the EC modulus in words.

2.5.37.2.6 #define CC_ECPKI_ORDER_MAX_LENGTH_IN_WORDS (CC_ECPKI_MODUL_MAX_LENGTH_IN_WORDS + 1)

The maximal size of the EC order in words.

2.5.37.2.7 #define CC_PKA_BARRETT_MOD_TAG_BUFF_SIZE_IN_WORDS 5

The size of the buffers for Barrett modulus tag NP, used in PKI algorithms.

2.5.37.2.8 #define CC_PKA_BARRETT_MOD_TAG_SIZE_IN_WORDS (((CC_PKA_WORD_SIZE_IN_BITS + PKA_EXTRA_BITS - 1) + (CC_BITS_IN_32BIT_WORD - 1)) / CC_BITS_IN_32BIT_WORD)

Actual size of Barrett modulus tag NP in words for current HW platform

2.5.37.2.9 #define CC_PKA_DOMAIN_BUFF_SIZE_IN_WORDS (2*CC_PKA_BARRETT_MOD_TAG_BUFF_SIZE_IN_WORDS)

The maximal size of the EC domain in words.

2.5.37.2.10 #define CC_PKA_ECDH_BUFF_MAX_LENGTH_IN_WORDS (2*CC_ECPKI_ORDER_MAX_LENGTH_IN_WORDS + CC_PKA_ECPKI_SCALAR_MUL_BUFF_MAX_LENGTH_IN_WORDS)

The size of the ECC ECDH temporary buffer in words.

2.5.37.2.11 #define CC_PKA_ECDSA_NAF_BUFF_MAX_LENGTH_IN_WORDS (COUNT_NAF_WORDS_PER_KEY_WORD*CC_ECPKI_ORDER_MAX_LENGTH_IN_WORDS + 1)

The maximal length of the ECC NAF buffer.

2.5.37.2.12 #define CC_PKA_ECDSA_SIGN_BUFF_MAX_LENGTH_IN_WORDS (6*CC_ECPKI_MODUL_MAX_LENGTH_IN_WORDS+CC_PKA_ECPKI_SCALAR_MUL_BUFF_MAX_LENGTH_IN_WORDS)

The size of the ECC sign temporary buffer in words.

2.5.37.2.13 #define CC_PKA_ECDSA_VERIFY_BUFF_MAX_LENGTH_IN_WORDS (3*CC_ECPKI_MODUL_MAX_LENGTH_IN_WORDS)

The size of the ECC verify temporary buffer in words.

2.5.37.2.14 #define CC_PKA_ECPKI_BARRETT_MOD_TAG_BUFF_SIZE_IN_WORDS CC_PKA_BARRETT_MOD_TAG_BUFF_SIZE_IN_WORDS

The size of the buffers for Barrett modulus tag NP, used in ECC.

2.5.37.2.15 #define CC_PKA_ECPKI_BUILD_TMP_BUFF_MAX_LENGTH_IN_WORDS (3*CC_ECPKI_MODUL_MAX_LENGTH_IN_WORDS+CC_PKA_ECPKI_SCALAR_MUL_BUFF_MAX_LENGTH_IN_WORDS)

The size of the ECC temporary buffer in words.

2.5.37.2.16 #define CC_PKA_ECPKI_SCALAR_MUL_BUFF_MAX_LENGTH_IN_WORDS (CC_PKA_ECDSA_NAF_BUFF_MAX_LENGTH_IN_WORDS + CC_ECPKI_MODUL_MAX_LENGTH_IN_WORDS + 2)

The size of the Scalar buffer in words.

2.5.37.2.17 #define CC_PKA_KG_BUFF_MAX_LENGTH_IN_WORDS (2*CC_ECPKI_ORDER_MAX_LENGTH_IN_WORDS + CC_PKA_ECPKI_SCALAR_MUL_BUFF_MAX_LENGTH_IN_WORDS)

The size of the PKA KG temporary buffer in words.

2.5.37.2.18 #define CC_PKA_KGDATA_BUFF_SIZE_IN_WORDS (3*CC_PKA_BARRETT_MOD_TAG_BUFF_SIZE_IN_WORDS + 3*CC_PKA_MAXIMUM_MOD_BUFFER_SIZE_IN_WORDS)

The maximal size of the PKA KG buffer in words

2.5.37.2.19 #define CC_PKA_MAXIMUM_MOD_BUFFER_SIZE_IN_WORDS CC_RSA_MAXIMUM_MOD_BUFFER_SIZE_IN_WORDS

The maximal size of the PKA modulus.

2.5.37.2.20 #define CC_PKA_PRIV_KEY_BUFF_SIZE_IN_WORDS (2*CC_PKA_BARRETT_MOD_TAG_BUFF_SIZE_IN_WORDS)

The maximal size of the PKA private-key in words.

2.5.37.2.21 #define CC_PKA_PUB_KEY_BUFF_SIZE_IN_WORDS (2*CC_PKA_BARRETT_MOD_TAG_BUFF_SIZE_IN_WORDS)

The maximal size of the PKA public-key in words.

2.5.37.2.22 #define CC_PKA_WORD_SIZE_IN_BITS 128

The size of the PKA engine word.

2.5.37.2.23 #define CC_RSA_MAX_KEY_GENERATION_HW_SIZE_BITS 4096

The maximal supported size of key-generation in RSA in bits.

2.5.37.2.24 #define CC_RSA_MAX_VALID_KEY_SIZE_VALUE_IN_BITS 4096

The maximal supported size of modulus in RSA in bits.

2.5.37.2.25 #define CC_RSA_MAXIMUM_MOD_BUFFER_SIZE_IN_WORDS ((CC_RSA_MAX_VALID_KEY_SIZE_VALUE_IN_BITS + CC_PKA_WORD_SIZE_IN_BITS) / CC_BITS_IN_32BIT_WORD)

The maximal RSA modulus size.

2.5.37.2.26 #define COUNT_NAF_WORDS_PER_KEY_WORD 8

The ECC NAF buffer definitions.

2.5.37.2.27 #define PKA_EXTRA_BITS 8

The maximal count of extra bits in PKA operations.

2.5.37.2.28 #define PKA_MAX_COUNT_OF_PHYS_MEM_REGS 32

The number of memory registers in PKA operations.

2.5.37.2.29 #define RSA_PKA_BARRETT_MOD_TAG_BUFF_SIZE_IN_BYTES (RSA_PKA_BARRETT_MOD_TAG_BUFF_SIZE_IN_WORDS*CC_32BIT_WORD_SIZE)

Size of buffer for Barrett modulus tag in bytes.

2.5.37.2.30 #define RSA_PKA_BARRETT_MOD_TAG_BUFF_SIZE_IN_WORDS 5

Size of buffer for Barrett modulus tag in words.

2.5.38 Random number definitions

Contains all random number definitions.

2.5.38.1 Modules

- **CryptoCell random-number generation definitions.**

  Contains the CryptoCell random-number generation definitions.

- **CryptoCell random-number specific errors**

  Contains the definitions of the CryptoCell RND errors.

- **CryptoCell true-random-number generation definitions.**

  Contains the CryptoCell true-random-number generation defines.

2.5.39 SM2 APIs

Contains SM2 APIs and definitions.

2.5.39.1 Functions

- const **CCEcpkiDomain_t** ***CC_EcpkiGetSm2Domain** (void)

  The function returns the domain pointer of SM2.

2.5.39.2 Detailed description

Using Sign/Verify API is straightforward, just call Sign/Verify functions and provide the message hash that should be calculated by Sm2ComputeMessageDigest.

Use the key exchange APIs in the following order:

1. Both parties should first call to the **CC_Sm2KeyExchangeContext_init()** function.
2. Party A should call to **CC_Sm2CalculateECPoint()** and send the ephemeral public key to the party B.
3. After calling **CC_EcpkiPubKeyExport()** the ephemeral public key should be sent as a byte array.
4. The party B needs to verify that the ephemeral public key from party A is on the curve, by calling **CC_EcpkiPublKeyBuildAndCheck()** with checkmode=ECpublKeyPartlyCheck.
5. Party B in its order should call to **CC_Sm2CalculateECPoint()** and **CC_Sm2CalculateSharedSecret()** functions and send the ephemeral public key and, optionally the outside confirmation value to the party A.
6. The party A - calls **CC_Sm2CalculateSharedSecret()** and optionally sends to party B its outside confirmation value.
7. Each party may call the **CC_Sm2Confirmation()** function (if confirmation value was used in the previous steps).
8. In case of an agreement, each party calls the **CC_Sm2Kdf()** function in order to finally get the shared key.

2.5.39.3 Function documentation

2.5.39.3.1 const CCEcpkiDomain_t*CC_EcpkiGetSm2Domain (void)

Returns:

Domain pointer.

2.5.40 SM3 APIs

Contains SM3 APIs and definitions.

2.5.40.1 Modules

- **CryptoCell SM3 specific errors**

  Contains the definitions of the CryptoCell SM3 errors.

- **CryptoCell SM3 type definitions**

  Contains CryptoCell SM3 type definitions.

2.5.40.2 Functions

- **CIMPORT_C CCError_t CC_Sm3Init** (**CCSm3UserContext_t** *pContextID)

  This function initializes the SM3 machine and the SM3 Context.

- **CIMPORT_C CCError_t CC_Sm3Update** (**CCSm3UserContext_t** *pContextID, uint8_t *pDataIn, size_t DataInSize)

  This function processes a block of data to be HASHed.

- **CIMPORT_C CCError_t CC_Sm3Finish** (**CCSm3UserContext_t** *pContextID, **CCSm3ResultBuf_t** Sm3ResultBuff)

  This function finalize the process of SM3 data block.

- **CIMPORT_C CCError_t CC_Sm3Free** (**CCSm3UserContext_t** *pContextID)

  This function frees the context if the operation had failed.

- **CIMPORT_C CCError_t CC_Sm3** (uint8_t *pDataIn, size_t DataInSize, **CCSm3ResultBuf_t** Sm3ResultBuff)

  This function provides an SM3 function to process one buffer of data.

2.5.40.3 Function documentation

2.5.40.3.1 CIMPORT_C CCError_t **CC_Sm3 (uint8_t * *pDataIn*, size_t *DataInSize*,** CCSm3ResultBuf_t ***Sm3ResultBuff*)**

The function allocates an internal SM3 Context, and initializes it with the cryptographic attributes that are needed for the SM3 block operation (initialize H's value for the SM3 algorithm). Then it processes the data block, calculating the SM3 hash. Finally, it returns the data buffer's message digest.

Parameters:

Parameter	Description
pDataIn	Pointer to the buffer that stores the data to be hashed.
DataInSize	The size of the data to be hashed in bytes.

Return values:

Error code	Description
Sm3ResultBuff	Pointer to the result buffer for the the message digest.

Returns:

`CC_OK` on success.

A non-zero value from **cc_sm3_error.h** on failure.

2.5.40.3.2 CIMPORT_C CCError_t **CC_Sm3Finish (**CCSm3UserContext_t *** *pContextID,*** CCSm3ResultBuf_t ***Sm3ResultBuff*)**

It receives a handle to the SM3 Context, which was previously initialized by **CC_Sm3Init()** or by **CC_Sm3Update()**. It "adds" a header to the data block according to the relevant SM3 standard, and computes the final message digest.

Parameters:

Parameter	Description
`pContextID`	Pointer to the SM3 context buffer.

Return values:

Error code	Description
`Sm3ResultBuff`	Pointer to the result buffer for the the message digest.

Returns:

`CC_OK` on success.

A non-zero value from **cc_sm3_error.h** on failure.

2.5.40.3.3 CIMPORT_C CCError_t **CC_Sm3Free (**CCSm3UserContext_t *** *pContextID*)**

Parameters:

Parameter	Description
`pContextID`	Pointer to the SM3 context buffer.

Returns:

`CC_OK` on success

A non-zero value from **cc_sm3_error.h** on failure.

2.5.40.3.4 CIMPORT_C CCError_t **CC_Sm3Init (**CCSm3UserContext_t *** *pContextID*)**

It receives a pointer to SM3 context, and initializes it with the cryptographic attributes that are needed for the SM3 block operation (initializes H's value for the SM3 algorithm).

Parameters:

Parameter	Description
`pContextID`	Pointer to the SM3 context buffer. (allocated by the user)

Returns:

`CC_OK` on success.

A non-zero value from **cc_sm3_error.h** on failure.

2.5.40.3.5 CIMPORT_C CCError_t **CC_Sm3Update (**CCSm3UserContext_t *** *pContextID*, uint8_t * *pDataIn*, size_t *DataInSize*)**

It updates a SM3 Context that was previously initialized by **CC_Sm3Init()** or updated by a previous call to **CC_Sm3Update()**.

Parameters:

Parameter	Description
`pContextID`	Pointer to the SM3 context buffer. (allocated by the user)
`pDataIn`	Pointer to the buffer that stores the data to be hashed.
`DataInSize`	The size of the data to be hashed in bytes.

Returns:

`CC_OK` on success.

A non-zero value from **cc_sm3_error.h** on failure.

2.5.41 SM4 APIs

Contains SM4 APIs and definitions.

2.5.41.1 Modules

- **CryptoCell SM4 specific errors**

  Contains the definitions of the CryptoCell SM4 errors.

- **CryptoCell SM4 type definitions**

  Contains CryptoCell SM4 type definitions.

2.5.41.2 Functions

- **CIMPORT_C CCError_t CC_Sm4Init** (**CCSm4UserContext_t** *pContext, **CCSm4EncryptMode_t** encryptDecryptFlag, **CCSm4OperationMode_t** operationMode)

  This function is used to initialize a SM4 operation context. To operate the SM4 machine, this must be the first API called.

- **CIMPORT_C CCError_t CC_Sm4SetKey** (**CCSm4UserContext_t** *pContext, **CCSm4Key_t** pKey)

  This function sets the key information for the SM4 operation, in the context that was initialized by **CC_Sm4Init()**.

- **CIMPORT_C CCError_t CC_Sm4SetIv** (**CCSm4UserContext_t** *pContext, **CCSm4Iv_t** pIV)

  This function sets the IV or counter data for the following SM4 operations on the same context. The context must be first initialized by **CC_Sm4Init()**. It must be called at least once prior to the first **CC_Sm4Block()** operation on the same context - for those ciphers that require it. If needed, it can also be called to override the IV in the middle of a sequence of **CC_Sm4Block()** operations.

- **CIMPORT_C CCError_t CC_Sm4GetIv** (**CCSm4UserContext_t** *pContext, **CCSm4Iv_t** pIV)

  This function retrieves the current IV or counter data from the SM4 context.

- **CIMPORT_C CCError_t CC_Sm4Block** (**CCSm4UserContext_t** *pContext, uint8_t *pDataIn, size_t dataSize, uint8_t *pDataOut)

  This function performs a SM4 operation on an input data buffer, according to the configuration defined in the context parameter. It can be called as many times as needed, until all the input data is processed. The functions **CC_Sm4Init()**, **CC_Sm4SetKey()**, and for some ciphers **CC_Sm4SetIv()**, must be called before the first call to this API with the same context.

- **CIMPORT_C CCError_t CC_Sm4Finish** (**CCSm4UserContext_t** *pContext, uint8_t *pDataIn, size_t dataSize, uint8_t *pDataOut)

  This function is used to finish SM4 operation. It processes the last data block if needed, and finalizes the SM4 operation (cipher-specific).

- **CIMPORT_C CCError_t CC_Sm4Free** (**CCSm4UserContext_t** *pContext)

  This function releases and clears resources after SM4 operations.

- **CIMPORT_C CCError_t CC_Sm4** (**CCSm4Iv_t** pIV, **CCSm4Key_t** pKey, **CCSm4EncryptMode_t** encryptDecryptFlag, **CCSm4OperationMode_t** operationMode, uint8_t *pDataIn, size_t dataSize, uint8_t *pDataOut)

  This function performs a SM4 operation with a given key in a single call for all SM4 supported modes, and can be used when all data is available at the beginning of the operation.

2.5.41.3 Function documentation

2.5.41.3.1 CIMPORT_C CCError_t **CC_Sm4 (**CCSm4Iv_t ***pIV*, **CCSm4Key_t ***pKey*, **CCSm4EncryptMode_t ***encryptDecryptFlag*, **CCSm4OperationMode_t ***operationMode*, **uint8_t * *pDataIn*, size_t *dataSize*, uint8_t * *pDataOut*)**

Returns:

CC_OK on success,

A non-zero value from **cc_sm4_error.h** on failure.

Parameters:

I/O	Parameter	Description
in	pIV	Pointer to the buffer of the IV or counter. • For CBC mode - the IV value. • For CTR mode - the counter.
in	pKey	Pointer to the key data struct to be used for the SM4 operation. Must be 128bit.

I/O	Parameter	Description
in	encryptDecryptFlag	A flag specifying whether an SM4 Encrypt (CC_SM4_ENCRYPT) or Decrypt (CC_SM4_DECRYPT) operation should be performed.
in	operationMode	The operation cipher/mode: ECB / CBC / CTR.
in	pDataIn	Pointer to the buffer of the input data to the SM4. The pointer does not need to be aligned. For TZ, the size of the scatter/gather list representing the data buffer is limited to 128 entries, and the size of each entry is limited to 64KB (fragments larger than 64KB are broken into fragments <= 64KB).
in	dataSize	Size of the input data in bytes. For all modes must be >0, and a multiple of 16 bytes.
out	pDataOut	Pointer to the output buffer. The pointer does not need to be aligned. For TZ, the size of the scatter/gather list representing the data buffer is limited to 128 entries, and the size of each entry is limited to 64KB (fragments larger than 64KB are broken into fragments <= 64KB).

2.5.41.3.2 CIMPORT_C CCError_t **CC_Sm4Block** (CCSm4UserContext_t * ***pContext***, uint8_t * ***pDataIn***, size_t ***dataSize***, uint8_t * ***pDataOut***)

Returns:

CC_OK on success,

A non-zero value from **cc_sm4_error.h** on failure.

Parameters:

I/O	Parameter	Description
in	pContext	Pointer to the SM4 context.
in	pDataIn	Pointer to the buffer of the input data to the SM4. The pointer does not need to be aligned. For TZ, the size of the scatter/gather list representing the data buffer is limited to 128 entries, and the size of each entry is limited to 64KB (fragments larger than 64KB are broken into fragments <= 64KB).
in	dataSize	Size of the input data in bytes. For all modes must be >0, and a multiple of 16 bytes.
out	pDataOut	Pointer to the output buffer. The pointer does not need to be aligned. For TZ, the size of the scatter/gather list representing the data buffer is limited to 128 entries, and the size of each entry is limited to 64KB (fragments larger than 64KB are broken into fragments <= 64KB).

2.5.41.3.3 CIMPORT_C CCError_t **CC_Sm4Finish** (CCSm4UserContext_t * ***pContext***, uint8_t * ***pDataIn***, size_t ***dataSize***, uint8_t * ***pDataOut***)

Returns:

CC_OK on success,

A non-zero value from **cc_sm4_error.h** on failure.

Parameters:

I/O	Parameter	Description
in	`pContext`	Pointer to the SM4 context.
in	`pDataIn`	Pointer to the buffer of the input data to the SM4. The pointer does not need to be aligned. For TZ, the size of the scatter/gather list representing the data buffer is limited to 128 entries, and the size of each entry is limited to 64KB (fragments larger than 64KB are broken into fragments <= 64KB).
in	`dataSize`	The size of the input data in bytes. can be 0. For ECB, CBC modes MUST be a multiple of 16 bytes.
out	`pDataOut`	Pointer to the output buffer. The pointer does not need to be aligned. For TZ, the size of the scatter/gather list representing the data buffer is limited to 128 entries, and the size of each entry is limited to 64KB (fragments larger than 64KB are broken into fragments <= 64KB).

2.5.41.3.4 CIMPORT_C CCError_t **CC_Sm4Free (**CCSm4UserContext_t * ***pContext*)**

Returns:

`CC_OK` on success,

A non-zero value from **cc_sm4_error.h** on failure.

Parameters:

I/O	Parameter	Description
in	`pContext`	Pointer to the SM4 context.

2.5.41.3.5 CIMPORT_C CCError_t **CC_Sm4GetIv (**CCSm4UserContext_t * ***pContext*,** CCSm4Iv_t ***pIV*)**

Returns:

`CC_OK` on success,

A non-zero value from **cc_sm4_error.h** on failure.

Parameters:

I/O	Parameter	Description
in	`pContext`	Pointer to the SM4 context.
out	`pIV`	Pointer to the buffer of the IV or counter. • For CBC mode - the IV value. • For CTR mode - the counter.

2.5.41.3.6 CIMPORT_C CCError_t **CC_Sm4Init (**CCSm4UserContext_t * ***pContext*,** CCSm4EncryptMode_t ***encryptDecryptFlag*,** CCSm4OperationMode_t ***operationMode*)**

Returns:

`CC_OK` on success,

A non-zero value from **cc_sm4_error.h** on failure.

Parameters:

I/O	Parameter	Description
in	pContext	Pointer to the SM4 context buffer that is allocated by the caller and initialized by this API. Should be used in all subsequent calls that are part of the same operation.
in	encryptDecryptFlag	A flag specifying whether an SM4 Encrypt (CC_SM4_ENCRYPT) or Decrypt (CC_SM4_DECRYPT) operation should be performed.
in	operationMode	The operation cipher/mode: ECB / CBC / CTR.

2.5.41.3.7 CIMPORT_C CCError_t **CC_Sm4SetIv (**CCSm4UserContext_t * ***pContext,*** CCSm4Iv_t ***pIV)***

Returns:

CC_OK on success,

A non-zero value from **cc_sm4_error.h** on failure.

Parameters:

I/O	Parameter	Description
in	pContext	Pointer to the SM4 context.
in	pIV	Pointer to the buffer of the IV, counter or tweak. • For CBC mode - the IV value. • For CTR mode - the counter

2.5.41.3.8 CIMPORT_C CCError_t **CC_Sm4SetKey (**CCSm4UserContext_t * ***pContext,*** CCSm4Key_t ***pKey)***

Returns:

CC_OK on success,

A non-zero value from **cc_sm4_error.h** on failure.

Parameters:

I/O	Parameter	Description
in	pContext	Pointer to the SM4 context, after it was initialized by **CC_Sm4Init()**.
in	pKey	Pointer to the key data struct to be used for the SM4 operation. Must be 128bit.

2.5.42 Specific errors of the CryptoCell PAL APIs

Contains platform-dependent PAL-API error definitions.

2.5.42.1 Macros

- #define **CC_PAL_BASE_ERROR** 0x0F000000
- #define **CC_PAL_MEM_BUF1_GREATER CC_PAL_BASE_ERROR** + 0x01UL

- #define **CC_PAL_MEM_BUF2_GREATER CC_PAL_BASE_ERROR** + 0x02UL
- #define **CC_PAL_SEM_CREATE_FAILED CC_PAL_BASE_ERROR** + 0x03UL
- #define **CC_PAL_SEM_DELETE_FAILED CC_PAL_BASE_ERROR** + 0x04UL
- #define **CC_PAL_SEM_WAIT_TIMEOUT CC_PAL_BASE_ERROR** + 0x05UL
- #define **CC_PAL_SEM_WAIT_FAILED CC_PAL_BASE_ERROR** + 0x06UL
- #define **CC_PAL_SEM_RELEASE_FAILED CC_PAL_BASE_ERROR** + 0x07UL
- #define **CC_PAL_ILLEGAL_ADDRESS CC_PAL_BASE_ERROR** + 0x08UL

2.5.42.2 Macro definition documentation

2.5.42.2.1 #define CC_PAL_BASE_ERROR 0x0F000000

The PAL error base.

2.5.42.2.2 #define CC_PAL_ILLEGAL_ADDRESS CC_PAL_BASE_ERROR + 0x08UL

Illegal PAL address.

2.5.42.2.3 #define CC_PAL_MEM_BUF1_GREATER CC_PAL_BASE_ERROR + 0x01UL

Buffer 1 is greater than buffer 2 error.

2.5.42.2.4 #define CC_PAL_MEM_BUF2_GREATER CC_PAL_BASE_ERROR + 0x02UL

Buffer 2 is greater than buffer 1 error.

2.5.42.2.5 #define CC_PAL_SEM_CREATE_FAILED CC_PAL_BASE_ERROR + 0x03UL

Semaphore creation failed.

2.5.42.2.6 #define CC_PAL_SEM_DELETE_FAILED CC_PAL_BASE_ERROR + 0x04UL

Semaphore deletion failed.

2.5.42.2.7 #define CC_PAL_SEM_RELEASE_FAILED CC_PAL_BASE_ERROR + 0x07UL

Semaphore release failed.

2.5.42.2.8 #define CC_PAL_SEM_WAIT_FAILED CC_PAL_BASE_ERROR + 0x06UL

Semaphore wait failed.

2.5.42.2.9 #define CC_PAL_SEM_WAIT_TIMEOUT CC_PAL_BASE_ERROR + 0x05UL

Semaphore reached timeout.

2.5.43 Specific errors of the CryptoCell utility module APIs

Contains utility API error definitions.

2.5.43.1 Macros

- #define CC_UTIL_OK 0x00UL
- #define CC_UTIL_MODULE_ERROR_BASE 0x80000000
- #define CC_UTIL_INVALID_KEY_TYPE (CC_UTIL_MODULE_ERROR_BASE + 0x00UL)
- #define CC_UTIL_DATA_IN_POINTER_INVALID_ERROR (CC_UTIL_MODULE_ERROR_BASE + 0x01UL)
- #define CC_UTIL_DATA_IN_SIZE_INVALID_ERROR (CC_UTIL_MODULE_ERROR_BASE + 0x02UL)
- #define CC_UTIL_DATA_OUT_POINTER_INVALID_ERROR (CC_UTIL_MODULE_ERROR_BASE + 0x03UL)
- #define CC_UTIL_DATA_OUT_SIZE_INVALID_ERROR (CC_UTIL_MODULE_ERROR_BASE + 0x04UL)
- #define CC_UTIL_FATAL_ERROR (CC_UTIL_MODULE_ERROR_BASE + 0x05UL)
- #define CC_UTIL_ILLEGAL_PARAMS_ERROR (CC_UTIL_MODULE_ERROR_BASE + 0x06UL)
- #define CC_UTIL_BAD_ADDR_ERROR (CC_UTIL_MODULE_ERROR_BASE + 0x07UL)
- #define CC_UTIL_EK_DOMAIN_INVALID_ERROR (CC_UTIL_MODULE_ERROR_BASE + 0x08UL)
- #define CC_UTIL_KDR_INVALID_ERROR (CC_UTIL_MODULE_ERROR_BASE + 0x09UL)
- #define CC_UTIL_KCP_INVALID_ERROR (CC_UTIL_MODULE_ERROR_BASE + 0x0AUL)
- #define CC_UTIL_KPICV_INVALID_ERROR (CC_UTIL_MODULE_ERROR_BASE + 0x0BUL)
- #define CC_UTIL_KCST_NOT_DISABLED_ERROR (CC_UTIL_MODULE_ERROR_BASE + 0x0CUL)
- #define CC_UTIL_LCS_INVALID_ERROR (CC_UTIL_MODULE_ERROR_BASE + 0x0DUL)
- #define CC_UTIL_SESSION_KEY_ERROR (CC_UTIL_MODULE_ERROR_BASE + 0x0EUL)
- #define CC_UTIL_INVALID_USER_KEY_SIZE (CC_UTIL_MODULE_ERROR_BASE + 0x0FUL)
- #define CC_UTIL_ILLEGAL_LCS_FOR_OPERATION_ERR (CC_UTIL_MODULE_ERROR_BASE + 0x10UL)
- #define CC_UTIL_INVALID_PRF_TYPE (CC_UTIL_MODULE_ERROR_BASE + 0x11UL)
- #define CC_UTIL_INVALID_HASH_MODE (CC_UTIL_MODULE_ERROR_BASE + 0x12UL)
- #define CC_UTIL_UNSUPPORTED_HASH_MODE (CC_UTIL_MODULE_ERROR_BASE + 0x13UL)
- #define CC_UTIL_KEY_UNUSABLE_ERROR (CC_UTIL_MODULE_ERROR_BASE + 0x14UL)

- #define **CC_UTIL_PM_ERROR** (**CC_UTIL_MODULE_ERROR_BASE** + 0x15UL)
- #define **CC_UTIL_SD_IS_SET_ERROR** (**CC_UTIL_MODULE_ERROR_BASE** + 0x16UL)

2.5.43.2 typedefs

- typedef uint32_t **CCUtilError_t**

2.5.43.3 Macro definition documentation

2.5.43.3.1 #define CC_UTIL_BAD_ADDR_ERROR (CC_UTIL_MODULE_ERROR_BASE + 0x07UL)

Invalid address given.

2.5.43.3.2 #define CC_UTIL_DATA_IN_POINTER_INVALID_ERROR (CC_UTIL_MODULE_ERROR_BASE + 0x01UL)

Illegal data-in pointer.

2.5.43.3.3 #define CC_UTIL_DATA_IN_SIZE_INVALID_ERROR (CC_UTIL_MODULE_ERROR_BASE + 0x02UL)

Illegal data-in size.

2.5.43.3.4 #define CC_UTIL_DATA_OUT_POINTER_INVALID_ERROR (CC_UTIL_MODULE_ERROR_BASE + 0x03UL)

Illegal data-out pointer.

2.5.43.3.5 #define CC_UTIL_DATA_OUT_SIZE_INVALID_ERROR (CC_UTIL_MODULE_ERROR_BASE + 0x04UL)

Illegal data-out size.

2.5.43.3.6 #define CC_UTIL_EK_DOMAIN_INVALID_ERROR (CC_UTIL_MODULE_ERROR_BASE + 0x08UL)

Illegal domain for endorsement key.

2.5.43.3.7 #define CC_UTIL_FATAL_ERROR (CC_UTIL_MODULE_ERROR_BASE + 0x05UL)

Fatal error.

2.5.43.3.8 #define CC_UTIL_ILLEGAL_LCS_FOR_OPERATION_ERR (CC_UTIL_MODULE_ERROR_BASE + 0x10UL)

Illegal LCS for the required operation.

2.5.43.3.9 #define CC_UTIL_ILLEGAL_PARAMS_ERROR (CC_UTIL_MODULE_ERROR_BASE + 0x06UL)

Illegal parameters.

2.5.43.3.10 #define CC_UTIL_INVALID_HASH_MODE (CC_UTIL_MODULE_ERROR_BASE + 0x12UL)

Invalid hash mode.

2.5.43.3.11 #define CC_UTIL_INVALID_KEY_TYPE (CC_UTIL_MODULE_ERROR_BASE + 0x00UL)

Illegal key type.

2.5.43.3.12 #define CC_UTIL_INVALID_PRF_TYPE (CC_UTIL_MODULE_ERROR_BASE + 0x11UL)

Invalid PRF type.

2.5.43.3.13 #define CC_UTIL_INVALID_USER_KEY_SIZE (CC_UTIL_MODULE_ERROR_BASE + 0x0FUL)

Illegal user key size.

2.5.43.3.14 #define CC_UTIL_KCP_INVALID_ERROR (CC_UTIL_MODULE_ERROR_BASE + 0x0AUL)

KCP is not valid.

2.5.43.3.15 #define CC_UTIL_KCST_NOT_DISABLED_ERROR (CC_UTIL_MODULE_ERROR_BASE + 0x0CUL)

KCST is not disabled.

2.5.43.3.16 #define CC_UTIL_KDR_INVALID_ERROR (CC_UTIL_MODULE_ERROR_BASE + 0x09UL)

HUK is not valid.

2.5.43.3.17 #define CC_UTIL_KEY_UNUSABLE_ERROR (CC_UTIL_MODULE_ERROR_BASE + 0x14UL)

Key is unusable.

2.5.43.3.18 #define CC_UTIL_KPICV_INVALID_ERROR (CC_UTIL_MODULE_ERROR_BASE + 0x0BUL)

KPICV is not valid.

2.5.43.3.19 #define CC_UTIL_LCS_INVALID_ERROR (CC_UTIL_MODULE_ERROR_BASE + 0x0DUL)

LCS is not valid.

2.5.43.3.20 #define CC_UTIL_MODULE_ERROR_BASE 0x80000000

The error base address definition.

2.5.43.3.21 #define CC_UTIL_OK 0x00UL

Success definition.

2.5.43.3.22 #define CC_UTIL_PM_ERROR (CC_UTIL_MODULE_ERROR_BASE + 0x15UL)

Power Management error.

2.5.43.3.23 #define CC_UTIL_SD_IS_SET_ERROR (CC_UTIL_MODULE_ERROR_BASE + 0x16UL)

Security disable bit is asserted , API should not be used.

2.5.43.3.24 #define CC_UTIL_SESSION_KEY_ERROR (CC_UTIL_MODULE_ERROR_BASE + 0x0EUL)

Session key is not valid.

2.5.43.3.25 #define CC_UTIL_UNSUPPORTED_HASH_MODE (CC_UTIL_MODULE_ERROR_BASE + 0x13UL)

Unsupported hash mode.

2.5.43.4 typedef documentation

2.5.43.4.1 typedef uint32_t CCUtilError_t

Util Error type.

2.5.44 TRNG API definition

Contains API and definitions for generating TRNG buffer in full entropy mode.

2.5.44.1 Macros

- #define **CC_TRNG_MIN_ENTROPY_SIZE** 0
- #define **CC_TRNG_MAX_ENTROPY_SIZE** 8192

2.5.44.2 Functions

- **CCError_t CC_TrngEntropyGet** (size_t entropySizeBits, uint8_t *pOutEntropy, size_t outEntropySizeBytes)

  The function returns an entropy buffer in the requested size.

2.5.44.3 Macro definition documentation

2.5.44.3.1 #define CC_TRNG_MAX_ENTROPY_SIZE 8192

Maximal entropy size in bits.

2.5.44.3.2 #define CC_TRNG_MIN_ENTROPY_SIZE 0

Minimum entropy size in bits.

2.5.44.4 Function documentation

2.5.44.4.1 CCError_t CC_TrngEntropyGet (size_t *entropySizeBits*, uint8_t * *pOutEntropy*, size_t *outEntropySizeBytes*)

Returns:

`CC_OK` on success.

A non-zero value from **cc_trng_error.h** on failure.

Parameters:

I/O	Parameter	Description
in	`entropySizeBits`	The required entropy size in bits.Size must be bigger than CC_TRNG_MIN_ENTROPY_SIZE, and smaller than CC_TRNG_MAX_ENTROPY_SIZE.
out	`pOutEntropy`	Pointer to the entropy buffer.
in	`outEntropySizeBytes`	The entropy buffer size in bytes. The size must be big enough to hold the required entropySizeBits.

2.5.45 TRNG APIs

Contains TRNG APIs.

2.5.45.1 Modules

- **CryptoCell TRNG specific errors**

  Contains the definitions of the CryptoCell TRNG errors.

- **Random number definitions**

  Contains all random number definitions.

- **TRNG API definition**

  Contains API and definitions for generating TRNG buffer in full entropy mode.

2.5.46 bit-field operations macros

Contains bit-field operation macros.

2.6 Data Structure Documentation

2.6.1 CC_PalTrngParams_t struct reference

```
#include <cc_pal_trng.h>
```

2.6.1.1 Data Fields

- uint32_t **SubSamplingRatio1**
- uint32_t **SubSamplingRatio2**
- uint32_t **SubSamplingRatio3**
- uint32_t **SubSamplingRatio4**

2.6.1.2 Detailed description

Definition for the structure of the random-generator parameters of CryptoCell, containing the user-given parameters.

2.6.1.3 Field documentation

2.6.1.3.1 uint32_t CC_PalTrngParams_t::SubSamplingRatio1

The sampling ratio of ROSC #1.

2.6.1.3.2 uint32_t CC_PalTrngParams_t::SubSamplingRatio2

The sampling ratio of ROSC #2.

2.6.1.3.3 uint32_t CC_PalTrngParams_t::SubSamplingRatio3

The sampling ratio of ROSC #3.

2.6.1.3.4 uint32_t CC_PalTrngParams_t::SubSamplingRatio4

The sampling ratio of ROSC #4.

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

- cc_pal_trng.h

2.6.2 CC_Sm2KeContext_t struct reference

```
#include <cc_sm2.h>
```

2.6.2.1 Data Fields

- int **isInitiator**
- uint8_t **confirmation**
- **CCEcpkiUserPublKey_t pubKey**
- **CCEcpkiUserPrivKey_t privKey**
- **CCEcpkiUserPublKey_t remotePubKey**
- **CCEcpkiPointAffine_t ephemeral_pub**
- size_t **eph_pub_key_size**
- **CCEcpkiPointAffine_t remote_ephemeral_pub**
- size_t **remote_eph_pub_key_size**
- const char ***pId**
- size_t **idlen**
- const char ***pRemoteId**
- size_t **remoteIdLen**
- uint32_t **t** [**CC_ECPKI_MODUL_MAX_LENGTH_IN_WORDS**]
- **CCEcpkiPointAffine_t V**
- uint8_t **conf_value** [CC_SM3_RESULT_SIZE_IN_BYTES]
- size_t **conf_value_size**
- uint8_t **Z** [CC_SM3_RESULT_SIZE_IN_BYTES]
- uint8_t **Z_remote** [CC_SM3_RESULT_SIZE_IN_BYTES]
- size_t **Z_value_size**

2.6.2.2 Detailed description

A structure to define key exchange context. All byte arrays in this structure are stored in the big endian byte ordering, and all word arrays are in the little endian byte and word ordering.

2.6.2.3 Field documentation

2.6.2.3.1 uint8_t CC_Sm2KeContext_t::conf_value[CC_SM3_RESULT_SIZE_IN_BYTES]

The internal confirmation value of this side calculated and stored if confirmation == 1 or confirmation == 3 in **CC_Sm2CalculateSharedSecret()** function.

2.6.2.3.2 size_t CC_Sm2KeContext_t::conf_value_size

Size of the confirmation value.

2.6.2.3.3 uint8_t CC_Sm2KeContext_t::confirmation

First bit encodes weather this party wants confirmation, second bit encodes the confirmation for other party for example 3 for both parts:

- 1 - Only this party wants confirmation,
- 2 - Only the other party wants confirmation.

2.6.2.3.4 size_t CC_Sm2KeContext_t::eph_pub_key_size

The size in bytes of the ephemeral public key of this party.

2.6.2.3.5 CCEcpkiPointAffine_t CC_Sm2KeContext_t::ephemeral_pub

The ephemeral public key of this party.

2.6.2.3.6 size_t CC_Sm2KeContext_t::idlen

The size in bytes of the ID of this party.

2.6.2.3.7 int CC_Sm2KeContext_t::isInitiator

A flag to define the initiator of the key exchange protocol.

2.6.2.3.8 const char*CC_Sm2KeContext_t::pId

Pointer to the ID of this party as string.

2.6.2.3.9 const char*CC_Sm2KeContext_t::pRemoteId

Pointer to the ID of the other party as string (remote ID).

2.6.2.3.10 CCEcpkiUserPrivKey_t CC_Sm2KeContext_t::privKey

The private key of this party.

2.6.2.3.11 CCEcpkiUserPublKey_t CC_Sm2KeContext_t::pubKey

The public key of this party.

2.6.2.3.12 size_t CC_Sm2KeContext_t::remote_eph_pub_key_size

The size in bytes of the ephemeral public key of other party.

2.6.2.3.13 CCEcpkiPointAffine_t CC_Sm2KeContext_t::remote_ephemeral_pub

The ephemeral public key of other party.

2.6.2.3.14 size_t CC_Sm2KeContext_t::remoteIdLen

The size in bytes of the ID of the other party.

2.6.2.3.15 CCEcpkiUserPublKey_t CC_Sm2KeContext_t::remotePubKey

The public key of the other party.

2.6.2.3.16 uint32_t CC_Sm2KeContext_t::t[CC_ECPKI_MODUL_MAX_LENGTH_IN_WORDS]

t value calculated and stored in **CC_Sm2CalculateECPoint()** function.

2.6.2.3.17 CCEcpkiPointAffine_t CC_Sm2KeContext_t::V

The shared secret, V/U value (shared secret) calculated and stored in **CC_Sm2CalculateSharedSecret()** function.

2.6.2.3.18 uint8_t CC_Sm2KeContext_t::Z[CC_SM3_RESULT_SIZE_IN_BYTES]

ID digests of this party - calculated and stored in **CC_Sm2KeyExchangeContext_init()** function.

2.6.2.3.19 uint8_t CC_Sm2KeContext_t::Z_remote[CC_SM3_RESULT_SIZE_IN_BYTES]

ID digests of the other party - calculated and stored in **CC_Sm2KeyExchangeContext_init()** function.

2.6.2.3.20 size_t CC_Sm2KeContext_t::Z_value_size

Size of the ID digest

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

- **cc_sm2.h**

2.6.3 CCAesHwKeyData_t struct reference

```
#include <cc_aes_defs.h>
```

2.6.3.1 Data Fields

- size_t **slotNumber**

2.6.3.2 Detailed description

The AES HW key Data.

2.6.3.3 Field documentation

2.6.3.3.1 size_t CCAesHwKeyData_t::slotNumber

< Slot number.

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

- cc_aes_defs.h

2.6.4 CCAesUserContext_t struct reference

```
#include <cc_aes_defs.h>
```

2.6.4.1 Data Fields

- uint32_t **buff** [**CC_AES_USER_CTX_SIZE_IN_WORDS**]

2.6.4.2 Detailed description

The context prototype of the user.

The argument type that is passed by the user to the AES APIs. The context saves the state of the operation, and must be saved by the user until the end of the API flow.

2.6.4.3 Field documentation

2.6.4.3.1 uint32_t CCAesUserContext_t::buff[CC_AES_USER_CTX_SIZE_IN_WORDS]

< The context buffer for internal usage.

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

- cc_aes_defs.h

2.6.5 CCAesUserKeyData_t struct reference

```
#include <cc_aes_defs.h>
```

2.6.5.1 Data Fields

- uint8_t ***pKey**
- size_t **keySize**

2.6.5.2 Detailed description

The AES key data of the user.

2.6.5.3 Field documentation

2.6.5.3.1 size_t CCAesUserKeyData_t::keySize

The size of the key in bytes. Valid values:

- For XTS mode (if supportes): 32 bytes or 64 bytes, indicating the full size of the double key (2x128 or 2x256 bit).
- For XCBC-MAC mode: 16 bytes, as limited by the standard.
- For all other modes: 16 bytes, 24 bytes or 32 bytes.

2.6.5.3.2 uint8_t*CCAesUserKeyData_t::pKey

A pointer to the key.

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

- cc_aes_defs.h

2.6.6 CCAxiAceConst_t union reference

```
#include <cc_axi_ctrl.h>
```

2.6.6.1 Data Fields

- struct {

  uint32_t **ARDOMAIN**: 2

  uint32_t **AWDOMAIN**: 2

  uint32_t **ARBAR**: 2

  uint32_t **AWBAR**: 2

  uint32_t **ARSNOOP**: 4

  uint32_t **AWSNOOP_NOT_ALIGNED**: 3

  uint32_t **AWSNOOP_ALIGNED**: 3

  uint32_t **AWADDR_NOT_MASKED**: 7

  uint32_t **AWLEN_VAL**: 4

  } **bitField**
- uint32_t **word**

2.6.6.2 Detailed description

List ACE configuration for the Secure AXI transactions.

2.6.6.3 Field documentation

2.6.6.3.1 uint32_t CCAxiAceConst_t::ARBAR

ACE ARBAR constant value.

2.6.6.3.2 uint32_t CCAxiAceConst_t::ARDOMAIN

ACE ARDOMAIN constant value.

2.6.6.3.3 uint32_t CCAxiAceConst_t::ARSNOOP

ACE ARSNOOP constant value.

2.6.6.3.4 uint32_t CCAxiAceConst_t::AWADDR_NOT_MASKED

AWADDRESS not mask value.

2.6.6.3.5 uint32_t CCAxiAceConst_t::AWBAR

ACE AWBAR constant value.

2.6.6.3.6 uint32_t CCAxiAceConst_t::AWDOMAIN

ACE AWDOMAIN constant value.

2.6.6.3.7 uint32_t CCAxiAceConst_t::AWLEN_VAL

AWLEN value.

2.6.6.3.8 uint32_t CCAxiAceConst_t::AWSNOOP_ALIGNED

ACE AWSNOOP constant value when unaligned transaction is used.

2.6.6.3.9 uint32_t CCAxiAceConst_t::AWSNOOP_NOT_ALIGNED

ACE AWSNOOP constant value when unaligned transaction is used.

2.6.6.3.10 struct { ... } CCAxiAceConst_t::bitField

A bit field structure defining the ACE configuration.

2.6.6.3.11 uint32_t CCAxiAceConst_t::word

Reserved.

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

- cc_axi_ctrl.h

2.6.7 CCAxiFields_t struct reference

```
#include <cc_axi_ctrl.h>
```

2.6.7.1 Data Fields

- **CCAxiAceConst_t AXIM_ACE_CONST**
- **CCAximCacheParams_t AXIM_CACHE_PARAMS**

2.6.7.2 Detailed description

Structure holding the AXI configuration.

2.6.7.3 Field documentation

2.6.7.3.1 CCAxiAceConst_t **CCAxiFields_t::AXIM_ACE_CONST**

List ACE configuration for the Secure AXI transactions.

2.6.7.3.2 CCAximCacheParams_t **CCAxiFields_t::AXIM_CACHE_PARAMS**

AXI master configuration for DMA.

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

- cc_axi_ctrl.h

2.6.8 CCAximCacheParams_t union reference

```
#include <cc_axi_ctrl.h>
```

2.6.8.1 Data Fields

- struct {

  uint32_t **AWCACHE_LAST**: 4

  uint32_t **AWCACHE**: 4

  uint32_t **ARCACHE**: 4

  } **bitField**
- uint32_t **word**

2.6.8.2 Detailed description

AXI master configuration for DMA.

2.6.8.3 Field documentation

2.6.8.3.1 uint32_t CCAximCacheParams_t::ARCACHE

Configure the ARCACHE last transaction for DMA.

2.6.8.3.2 uint32_t CCAximCacheParams_t::AWCACHE

Configure the AWCACHE transaction for DMA.

2.6.8.3.3 uint32_t CCAximCacheParams_t::AWCACHE_LAST

Configure the AWCACHE last transaction for DMA.

2.6.8.3.4 struct { ... } CCAximCacheParams_t::bitField

A bit field structure defining the AXI master configuration.

2.6.8.3.5 uint32_t CCAximCacheParams_t::word

Reserved.

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

- cc_axi_ctrl.h

2.6.9 CCCertKatContext_t union reference

```
#include <cc_cert_ctx.h>
```

2.6.9.1 Data Fields

- CCRsaFipsKatContext_t **fipsRsaCtx**
- **CCEcdsaFipsKatContext_t fipsEcdsaCtx**
- CCDhFipsKat_t **fipsDhCtx**
- **CCEcdhFipsKatContext_t fipsEcdhCtx**
- CCPrngFipsKatCtx_t **fipsPrngCtx**
- **CCSm2FipsKatContext_t fipsSm2Ctx**

2.6.9.2 Detailed description

Definitions for the certification context.

2.6.9.3 Field documentation

2.6.9.3.1 CCDhFipsKat_t CCCertKatContext_t::fipsDhCtx

Definition for DH certification context.

2.6.9.3.2 CCEcdhFipsKatContext_t CCCertKatContext_t::fipsEcdhCtx

Definition for ECDH certification context.

2.6.9.3.3 CCEcdsaFipsKatContext_t CCCertKatContext_t::fipsEcdsaCtx

Definition for ECC certification context.

2.6.9.3.4 CCPrngFipsKatCtx_t CCCertKatContext_t::fipsPrngCtx

Definition for DRBG certification context.

2.6.9.3.5 CCRsaFipsKatContext_t CCCertKatContext_t::fipsRsaCtx

Definition for RSA certification context.

2.6.9.3.6 CCSm2FipsKatContext_t CCCertKatContext_t::fipsSm2Ctx

Definition for SM2 certification context.

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

- **cc_cert_ctx.h**

2.6.10 CCEcdhFipsKatContext_t struct reference

```
#include <cc_ecpki_types.h>
```

2.6.10.1 Data Fields

- **CCEcpkiUserPublKey_t pubKey**
- **CCEcpkiUserPrivKey_t privKey**
- union {

  CCEcpkiBuildTempData_t ecpkiTempData

  CCEcdhTempData_t ecdhTempBuff

  } **tmpData**
- uint8_t **secretBuff** [**CC_ECPKI_FIPS_ORDER_LENGTH**]

2.6.10.2 Detailed description

ECDH KAT data structures for FIPS certification.

2.6.10.3 Field documentation

2.6.10.3.1 CCEcpkiUserPrivKey_t **CCEcdhFipsKatContext_t::privKey**

The data of the private key.

2.6.10.3.2 CCEcpkiUserPublKey_t **CCEcdhFipsKatContext_t::pubKey**

The data of the public key.

2.6.10.3.3 uint8_t CCEcdhFipsKatContext_t::secretBuff[CC_ECPKI_FIPS_ORDER_LENGTH]

The buffer for the secret key.

2.6.10.3.4 union { ... } CCEcdhFipsKatContext_t::tmpData

Internal buffers.

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

- **cc_ecpki_types.h**

2.6.11 CCEcdhTempData_t struct reference

```
#include <cc_ecpki_types_common.h>
```

2.6.11.1 Data Fields

- uint32_t **ccEcdhIntBuff** [**CC_PKA_ECDH_BUFF_MAX_LENGTH_IN_WORDS**]

2.6.11.2 Detailed description

The type of the ECDH temporary data.

2.6.11.3 Field documentation

2.6.11.3.1 uint32_t CCEcdhTempData_t::ccEcdhIntBuff[CC_PKA_ECDH_BUFF_MAX_LENGTH_IN_WORDS]

Temporary buffers.

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

- **cc_ecpki_types_common.h**

2.6.12 CCEcdsaFipsKatContext_t struct reference

```
#include <cc_ecpki_types.h>
```

2.6.12.1 Data Fields

- union {

  struct {

  CCEcpkiUserPrivKey_t PrivKey

  CCEcdsaSignUserContext_t signCtx

  } userSignData
- struct {

  CCEcpkiUserPublKey_t PublKey

  union {

  CCEcdsaVerifyUserContext_t verifyCtx

  CCEcpkiBuildTempData_t tempData

  } buildOrVerify
- } userVerifyData
- } keyContextData
- uint8_t signBuff [2 *CC_ECPKI_FIPS_ORDER_LENGTH]

2.6.12.2 Detailed description

ECDSA KAT data structures for FIPS certification. The ECDSA KAT tests are defined for domain 256r1.

2.6.12.3 Field documentation

2.6.12.3.1 union { ... } CCEcdsaFipsKatContext_t::keyContextData

The data of the key.

2.6.12.3.2 uint8_t CCEcdsaFipsKatContext_t::signBuff[2 *CC_ECPKI_FIPS_ORDER_LENGTH]

Internal buffer.

2.6.12.3.3 struct { ... } CCEcdsaFipsKatContext_t::userSignData

The data of the private key.

2.6.12.3.4 struct { ... } CCEcdsaFipsKatContext_t::userVerifyData

The data of the public key.

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

- cc_ecpki_types.h

2.6.13 CCEcdsaSignUserContext_t struct reference

The context definition of the user for the signing operation.

```
#include <cc_ecpki_types.h>
```

2.6.13.1 Data Fields

- uint32_t **context_buff** [(sizeof(**EcdsaSignContext_t**)+3)/4]
- uint32_t **valid_tag**

2.6.13.2 Detailed description

This context saves the state of the operation, and must be saved by the user until the end of the API flow.

2.6.13.3 Field documentation

2.6.13.3.1 uint32_t CCEcdsaSignUserContext_t::context_buff[(sizeof(EcdsaSignContext_t)+3)/4]

The data of the signing process.

2.6.13.3.2 uint32_t CCEcdsaSignUserContext_t::valid_tag

The validation tag.

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

- **cc_ecpki_types.h**

2.6.14 CCEcdsaVerifyUserContext_t struct reference

The context definition of the user for the verification operation.

```
#include <cc_ecpki_types_common.h>
```

2.6.14.1 Data Fields

- uint32_t **context_buff** [(sizeof(**EcdsaVerifyContext_t**)+3)/4]
- uint32_t **valid_tag**

2.6.14.2 Detailed description

The context saves the state of the operation, and must be saved by the user until the end of the API flow.

2.6.14.3 Field documentation

2.6.14.3.1 uint32_t CCEcdsaVerifyUserContext_t::context_buff[(sizeof(EcdsaVerifyContext_t)+3)/4]

The data of the verification process.

2.6.14.3.2 uint32_t CCEcdsaVerifyUserContext_t::valid_tag

The validation tag.

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

- cc_ecpki_types_common.h

2.6.15 CCEciesTempData_t struct reference

```
#include <cc_ecpki_types_common.h>
```

2.6.15.1 Data Fields

- CCEcpkiUserPrivKey_t PrivKey
- CCEcpkiUserPublKey_t PublKey
- CCEcpkiUserPublKey_t ConvPublKey
- uint32_t zz [3 *CC_ECPKI_MODUL_MAX_LENGTH_IN_WORDS+1]
- union {

  CCEcpkiBuildTempData_t buildTempbuff

  CCEcpkiKgTempData_t KgTempBuff

  CCEcdhTempData_t DhTempBuff

  } tmp

2.6.15.2 Detailed description

The temporary data definition of the ECIES.

2.6.15.3 Field documentation

2.6.15.3.1 CCEcpkiUserPublKey_t CCEciesTempData_t::ConvPublKey

The public-key data used by conversion from Mbed TLS to CryptoCell.

2.6.15.3.2 CCEcpkiUserPrivKey_t CCEciesTempData_t::PrivKey

The data of the private key.

2.6.15.3.3 CCEcpkiUserPublKey_t **CCEciesTempData_t::PublKey**

The data of the public key.

2.6.15.3.4 union { ... } CCEciesTempData_t::tmp

Internal buffers.

2.6.15.3.5 uint32_t CCEciesTempData_t::zz[3 *CC_ECPKI_MODUL_MAX_LENGTH_IN_WORDS+1]

Internal buffer.

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

- **cc_ecpki_types_common.h**

2.6.16 CCEcpkiBuildTempData_t struct reference

```
#include <cc_ecpki_types_common.h>
```

2.6.16.1 Data Fields

- uint32_t **ccBuildTmpIntBuff** [**CC_PKA_ECPKI_BUILD_TMP_BUFF_MAX_LENGTH_IN_WORDS**]

2.6.16.2 Detailed description

EC build temporary data.

2.6.16.3 Field documentation

2.6.16.3.1 uint32_t CCEcpkiBuildTempData_t::ccBuildTmpIntBuff[CC_PKA_ECPKI_BUILD_TMP_BUFF_MAX_LENGTH_IN_WORDS]

Temporary buffers.

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

- **cc_ecpki_types_common.h**

2.6.17 CCEcpkiDomain_t struct reference

The structure containing the EC domain parameters in little-endian form.

```
#include <cc_ecpki_types_common.h>
```

2.6.17.1 Data Fields

- uint32_t **ecP** [**CC_ECPKI_MODUL_MAX_LENGTH_IN_WORDS**]
- uint32_t **ecA** [**CC_ECPKI_MODUL_MAX_LENGTH_IN_WORDS**]
- uint32_t **ecB** [**CC_ECPKI_MODUL_MAX_LENGTH_IN_WORDS**]
- uint32_t **ecR** [**CC_ECPKI_MODUL_MAX_LENGTH_IN_WORDS**+1]
- uint32_t **ecGx** [**CC_ECPKI_MODUL_MAX_LENGTH_IN_WORDS**]
- uint32_t **ecGy** [**CC_ECPKI_MODUL_MAX_LENGTH_IN_WORDS**]
- uint32_t **ecH**
- uint32_t **llfBuff** [**CC_PKA_DOMAIN_LLF_BUFF_SIZE_IN_WORDS**]
- uint32_t **modSizeInBits**
- uint32_t **ordSizeInBits**
- uint32_t **barrTagSizeInWords**
- **CCEcpkiDomainID_t DomainID**
- int8_t **name** [20]

2.6.17.2 Detailed description

EC equation: `Y^2` = `X^3` + `A*X` + `B` over prime field `GFp` .

2.6.17.3 Field documentation

2.6.17.3.1 uint32_t CCEcpkiDomain_t::barrTagSizeInWords

The size of each inserted Barret tag in words. 0 if not inserted.

2.6.17.3.2 CCEcpkiDomainID_t CCEcpkiDomain_t::DomainID

The EC Domain identifier.

2.6.17.3.3 uint32_t CCEcpkiDomain_t::ecA[CC_ECPKI_MODUL_MAX_LENGTH_IN_WORDS]

EC equation parameter A.

2.6.17.3.4 uint32_t CCEcpkiDomain_t::ecB[CC_ECPKI_MODUL_MAX_LENGTH_IN_WORDS]

EC equation parameter B.

2.6.17.3.5 uint32_t CCEcpkiDomain_t::ecGx[CC_ECPKI_MODUL_MAX_LENGTH_IN_WORDS]

EC cofactor EC_Cofactor_K. The coordinates of the EC base point generator in projective form.

2.6.17.3.6 uint32_t CCEcpkiDomain_t::ecGy[CC_ECPKI_MODUL_MAX_LENGTH_IN_WORDS]

EC cofactor EC_Cofactor_K. The coordinates of the EC base point generator in projective form.

2.6.17.3.7 uint32_t CCEcpkiDomain_t::ecH

EC cofactor EC_Cofactor_K. The coordinates of the EC base point generator in projective form.

2.6.17.3.8 uint32_t CCEcpkiDomain_t::ecP[CC_ECPKI_MODUL_MAX_LENGTH_IN_WORDS]

EC modulus: P.

2.6.17.3.9 uint32_t CCEcpkiDomain_t::ecR[CC_ECPKI_MODUL_MAX_LENGTH_IN_WORDS+1]

Order of generator.

2.6.17.3.10 uint32_t CCEcpkiDomain_t::llfBuff[CC_PKA_DOMAIN_LLF_BUFF_SIZE_IN_WORDS]

Specific fields that are used by the low-level functions.

2.6.17.3.11 uint32_t CCEcpkiDomain_t::modSizeInBits

The size of fields in bits.

2.6.17.3.12 int8_t CCEcpkiDomain_t::name[20]

Internal buffer.

2.6.17.3.13 uint32_t CCEcpkiDomain_t::ordSizeInBits

The size of the order in bits.

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

- **cc_ecpki_types_common.h**

2.6.18 CCEcpkiKgFipsContext_t struct reference

```
#include <cc_ecpki_types.h>
```

2.6.18.1 Data Fields

- union {

  CCEcdsaSignUserContext_t signCtx

  CCEcdsaVerifyUserContext_t verifyCtx

} **operationCtx**

- uint32_t **signBuff** [2 ***CC_ECPKI_ORDER_MAX_LENGTH_IN_WORDS**]

2.6.18.2 Detailed description

ECPKI data structures for FIPS certification.

2.6.18.3 Field documentation

2.6.18.3.1 union { ... } CCEcpkiKgFipsContext_t::operationCtx

Signing and verification data.

2.6.18.3.2 uint32_t CCEcpkiKgFipsContext_t::signBuff[2 *CC_ECPKI_ORDER_MAX_LENGTH_IN_WORDS]

Internal buffer.

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

- **cc_ecpki_types.h**

2.6.19 CCEcpkiKgTempData_t struct reference

```
#include <cc_ecpki_types_common.h>
```

2.6.19.1 Data Fields

- uint32_t **ccKGIntBuff** [**CC_PKA_KG_BUFF_MAX_LENGTH_IN_WORDS**]

2.6.19.2 Detailed description

The temporary data type of the ECPKI KG.

2.6.19.3 Field documentation

2.6.19.3.1 uint32_t CCEcpkiKgTempData_t::ccKGIntBuff[CC_PKA_KG_BUFF_MAX_LENGTH_IN_WORDS]

Internal buffer.

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

- **cc_ecpki_types_common.h**

2.6.20 CCEcpkiPointAffine_t struct reference

```
#include <cc_ecpki_types.h>
```

2.6.20.1 Data Fields

- uint32_t **x** [**CC_ECPKI_MODUL_MAX_LENGTH_IN_WORDS**]
- uint32_t **y** [**CC_ECPKI_MODUL_MAX_LENGTH_IN_WORDS**]

2.6.20.2 Detailed description

The structure containing the EC point in affine coordinates and little endian form.

2.6.20.3 Field documentation

2.6.20.3.1 uint32_t CCEcpkiPointAffine_t::x[CC_ECPKI_MODUL_MAX_LENGTH_IN_WORDS]

The X coordinate of the point.

2.6.20.3.2 uint32_t CCEcpkiPointAffine_t::y[CC_ECPKI_MODUL_MAX_LENGTH_IN_WORDS]

The Y coordinate of the point.

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

- **cc_ecpki_types.h**

2.6.21 CCEcpkiPrivKey_t struct reference

```
#include <cc_ecpki_types_common.h>
```

2.6.21.1 Data Fields

- uint32_t **PrivKey** [**CC_ECPKI_MODUL_MAX_LENGTH_IN_WORDS**+1]
- **CCEcpkiDomain_t domain**
- **CCEcpkiScaProtection_t scaProtection**

2.6.21.2 Detailed description

The structure containing the data of the private key.

2.6.21.3 Field documentation

2.6.21.3.1 CCEcpkiDomain_t CCEcpkiPrivKey_t::domain

The EC domain.

2.6.21.3.2 uint32_t CCEcpkiPrivKey_t::PrivKey[CC_ECPKI_MODUL_MAX_LENGTH_IN_WORDS+1]

The data of the private key.

2.6.21.3.3 CCEcpkiScaProtection_t **CCEcpkiPrivKey_t::scaProtection**

The SCA protection mode.

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

- **cc_ecpki_types_common.h**

2.6.22 CCEcpkiPublKey_t struct reference

```
#include <cc_ecpki_types_common.h>
```

2.6.22.1 Data Fields

- uint32_t **x** [**CC_ECPKI_MODUL_MAX_LENGTH_IN_WORDS**]
- uint32_t **y** [**CC_ECPKI_MODUL_MAX_LENGTH_IN_WORDS**]
- **CCEcpkiDomain_t domain**
- uint32_t **pointType**

2.6.22.2 Detailed description

The structure containing the public key in affine coordinates.

2.6.22.3 Field documentation

2.6.22.3.1 CCEcpkiDomain_t **CCEcpkiPublKey_t::domain**

The EC Domain.

2.6.22.3.2 uint32_t CCEcpkiPublKey_t::pointType

The point type.

2.6.22.3.3 uint32_t CCEcpkiPublKey_t::x[CC_ECPKI_MODUL_MAX_LENGTH_IN_WORDS]

The X coordinate of the public key.

2.6.22.3.4 uint32_t CCEcpkiPublKey_t::y[CC_ECPKI_MODUL_MAX_LENGTH_IN_WORDS]

The Y coordinate of the public key.

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

- **cc_ecpki_types_common.h**

2.6.23 CCEcpkiUserPrivKey_t struct reference

The user structure prototype of the EC private key.

```
#include <cc_ecpki_types_common.h>
```

2.6.23.1 Data Fields

- uint32_t **valid_tag**
- uint32_t **PrivKeyDbBuff** [(sizeof(**CCEcpkiPrivKey_t**)+3)/4]

2.6.23.2 Detailed description

This structure must be saved by the user. It is used as input to ECC functions, for example, CC_EcdsaSign().

2.6.23.3 Field documentation

2.6.23.3.1 uint32_t CCEcpkiUserPrivKey_t::PrivKeyDbBuff[(sizeof(CCEcpkiPrivKey_t)+3)/4]

The data of the private key.

2.6.23.3.2 uint32_t CCEcpkiUserPrivKey_t::valid_tag

The validation tag.

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

- **cc_ecpki_types_common.h**

2.6.24 CCEcpkiUserPublKey_t struct reference

The user structure prototype of the EC public key.

```
#include <cc_ecpki_types_common.h>
```

2.6.24.1 Data Fields

- uint32_t **valid_tag**
- uint32_t **PublKeyDbBuff** [(sizeof(**CCEcpkiPublKey_t**)+3)/4]

2.6.24.2 Detailed description

This structure must be saved by the user. It is used as input to ECC functions, for example, CC_EcdsaVerify().

2.6.24.3 Field documentation

2.6.24.3.1 uint32_t CCEcpkiUserPublKey_t::PublKeyDbBuff[(sizeof(CCEcpkiPublKey_t)+3)/4]

The data of the public key.

2.6.24.3.2 uint32_t CCEcpkiUserPublKey_t::valid_tag

The validation tag.

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

- cc_ecpki_types_common.h

2.6.25 CCHashUserContext_t struct reference

```
#include <cc_hash_defs.h>
```

2.6.25.1 Data Fields

- uint32_t **buff** [**CC_HASH_USER_CTX_SIZE_IN_WORDS**]

2.6.25.2 Detailed description

The context prototype of the user. The argument type that is passed by the user to the hash APIs. The context saves the state of the operation, and must be saved by the user until the end of the API flow.

2.6.25.3 Field documentation

2.6.25.3.1 uint32_t CCHashUserContext_t::buff[CC_HASH_USER_CTX_SIZE_IN_WORDS]

The internal buffer.

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

- cc_hash_defs.h

2.6.26 CCPalDmaBlockInfo_t struct reference

User buffer scatter information.

```
#include <cc_pal_dma.h>
```

2.6.26.1 Data Fields

- CCDmaAddr_t **blockPhysAddr**
- uint32_t **blockSize**

2.6.26.2 Field documentation

2.6.26.2.1 CCDmaAddr_t CCPalDmaBlockInfo_t::blockPhysAddr

The physical address of the user buffer.

2.6.26.2.2 uint32_t CCPalDmaBlockInfo_t::blockSize

The block size of the user buffer.

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

- cc_pal_dma.h

2.6.27 CCRndContext_t struct reference

```
#include <cc_rnd_common.h>
```

2.6.27.1 Data Fields

- void ***rndState**
- void ***entropyCtx**
- **CCRndGenerateVectWorkFunc_t rndGenerateVectFunc**

2.6.27.2 Detailed description

The definition of the RND context that includes the CryptoCell RND state structure, and a function pointer for the RND-generation function.

2.6.27.3 Field documentation

2.6.27.3.1 void*CCRndContext_t::entropyCtx

A pointer to the entropy context.

This pointer should be allocated and assigned before calling **CC_LibInit()**.

2.6.27.3.2 CCRndGenerateVectWorkFunc_t CCRndContext_t::rndGenerateVectFunc

A pointer to the user-given function for generation of a random vector.

2.6.27.3.3 void*CCRndContext_t::rndState

A pointer to the internal state of the RND.

This pointer should be allocated in a physical and contiguous memory, accessible to the CryptoCell DMA. This pointer should be allocated and assigned before calling **CC_LibInit()**.

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

- cc_rnd_common.h

2.6.28 CCRndState_t struct reference

The structure for the RND state. This includes internal data that must be saved by the user between boots.

```
#include <cc_rnd_common.h>
```

2.6.28.1 Data Fields

- uint32_t **Seed** [**CC_RND_SEED_MAX_SIZE_WORDS**]
- uint32_t **PreviousRandValue** [**CC_AES_CRYPTO_BLOCK_SIZE_IN_WORDS**]
- uint32_t **PreviousAdditionalInput** [**CC_RND_ADDITINAL_INPUT_MAX_SIZE_WORDS**+3]
- uint32_t **AdditionalInput** [**CC_RND_ADDITINAL_INPUT_MAX_SIZE_WORDS**+4]
- uint32_t **AddInputSizeWords**
- uint32_t **ReseedCounter**
- uint32_t **KeySizeWords**
- uint32_t **StateFlag**
- uint32_t **ValidTag**
- **CCTrngState_t trngState**

2.6.28.2 Field documentation

2.6.28.2.1 uint32_t CCRndState_t::AddInputSizeWords

The size of the additional input in words.

2.6.28.2.2 uint32_t CCRndState_t::AdditionalInput[CC_RND_ADDITINAL_INPUT_MAX_SIZE_WORDS+4]

The additional-input buffer.

2.6.28.2.3 uint32_t CCRndState_t::KeySizeWords

The key size according to security strength:

- 128 bits: 4 words.
- 256 bits: 8 words.

2.6.28.2.4 uint32_t CCRndState_t::PreviousAdditionalInput[CC_RND_ADDITINAL_INPUT_MAX_SIZE_WORDS+3]

The previous additional-input buffer.

2.6.28.2.5 uint32_t CCRndState_t::PreviousRandValue[CC_AES_CRYPTO_BLOCK_SIZE_IN_WORDS]

The previous random data, used for continuous test.

2.6.28.2.6 uint32_t CCRndState_t::ReseedCounter

The Reseed counter (32-bit active). Indicates the number of requests for entropy. since instantiation or reseeding.

2.6.28.2.7 uint32_t CCRndState_t::Seed[CC_RND_SEED_MAX_SIZE_WORDS]

The random-seed buffer.

2.6.28.2.8 uint32_t CCRndState_t::StateFlag

The state flag used internally in the code.

2.6.28.2.9 CCTrngState_t CCRndState_t::trngState

TRNG state

2.6.28.2.10 uint32_t CCRndState_t::ValidTag

The validation tag used internally in the code.

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

- cc_rnd_common.h

2.6.29 CCSm2FipsKatContext_t struct reference

```
#include <cc_sm2.h>
```

2.6.29.1 Data Fields

- uint8_t **workBuff** [2+**CC_SM2_MODULE_LENGTH_IN_BYTES** *4+**CC_SM2_ORDER_LENGTH_IN_BYTES** *2+**CERT_SM2_DEFAULT_INPUT_AND_ID_SIZE**]
- **CCRndGenerateVectWorkFunc_t f_rng**
- void ***p_rng**

2.6.29.2 Detailed description

SM2 self-test data structure for Chinese certification.

2.6.29.3 Field documentation

2.6.29.3.1 CCRndGenerateVectWorkFunc_t CCSm2FipsKatContext_t::f_rng

A pointer to DRBG function

2.6.29.3.2 void*CCSm2FipsKatContext_t::p_rng

A pointer to the random context - the input to f_rng.

2.6.29.3.3 uint8_t CCSm2FipsKatContext_t::workBuff[2+CC_SM2_MODULE_LENGTH_IN_BYTES *4+ CC_SM2_ORDER_LENGTH_IN_BYTES *2+CERT_SM2_DEFAULT_INPUT_AND_ID_SIZE]

The working buffer for **CC_Sm2ComputeMessageDigest**

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

- **cc_sm2.h**

2.6.30 CCSm2KeyGenCHCertContext_t struct reference

```
#include <cc_sm2.h>
```

2.6.30.1 Data Fields

- uint8_t **workBuff** [2+**CC_SM2_MODULE_LENGTH_IN_BYTES** *4+**CC_SM2_ORDER_LENGTH_IN_BYTES** *2+**CERT_SM2_DEFAULT_INPUT_AND_ID_SIZE**]

2.6.30.2 Detailed description

SM2 self-test data structure for certification.

2.6.30.3 Field documentation

2.6.30.3.1 uint8_t CCSm2KeyGenCHCertContext_t::workBuff[2+CC_SM2_MODULE_LENGTH_IN_BYTES *4+ CC_SM2_ORDER_LENGTH_IN_BYTES *2+CERT_SM2_DEFAULT_INPUT_AND_ID_SIZE]

The working buffer for **CC_Sm2ComputeMessageDigest**

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

- **cc_sm2.h**

2.6.31 CCSm3UserContext_t struct reference

```
#include <cc_sm3_defs.h>
```

2.6.31.1 Data Fields

- uint32_t **buff** [**CC_SM3_USER_CTX_SIZE_IN_WORDS**]

2.6.31.2 Detailed description

The context prototype of the user. The argument type that is passed by the user to the SM3 APIs. The context saves the state of the operation, and must be saved by the user until the end of the API flow.

2.6.31.3 Field documentation

2.6.31.3.1 uint32_t CCSm3UserContext_t::buff[CC_SM3_USER_CTX_SIZE_IN_WORDS]

The internal buffer.

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

- **cc_sm3_defs.h**

2.6.32 CCSm4UserContext_t struct reference

```
#include <cc_sm4_defs.h>
```

2.6.32.1 Data Fields

- uint32_t **buff** [**CC_SM4_USER_CTX_SIZE_IN_WORDS**]

2.6.32.2 Detailed description

The context prototype of the user.

The argument type that is passed by the user to the SM4 APIs. The context saves the state of the operation, and must be saved by the user till the end of the API flow.

2.6.32.3 Field documentation

2.6.32.3.1 uint32_t CCSm4UserContext_t::buff[CC_SM4_USER_CTX_SIZE_IN_WORDS]

The context buffer for internal usage.

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

- **cc_sm4_defs.h**

2.6.33 CCTrngParams_t struct reference

```
#include <cc_rnd_common_trng.h>
```

2.6.33.1 Data Fields

- CC_PalTrngParams_t **userParams**
- uint32_t **RoscsAllowed**
- uint32_t **SubSamplingRatio**

2.6.33.2 Detailed description

The CC Random Generator Parameters structure **CCTrngParams_t** - containing the user given parameters and characterization values.

2.6.33.3 Field documentation

2.6.33.3.1 uint32_t CCTrngParams_t::RoscsAllowed

Valid ring oscillator lengths: bits 0,1,2,3.

2.6.33.3.2 uint32_t CCTrngParams_t::SubSamplingRatio

Sampling interval: count of ring oscillator cycles between consecutive bits sampling.

2.6.33.3.3 CC_PalTrngParams_t CCTrngParams_t::userParams

User provided parameters

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

- cc_rnd_common_trng.h

2.6.34 CCTrngState_t struct reference

```
#include <cc_rnd_common_trng.h>
```

2.6.34.1 Data Fields

- uint32_t **LastTrngRosc**

2.6.34.2 Detailed description

The structure for the RND state. This includes internal data that must be saved by the user between boots.

2.6.34.3 Field documentation

2.6.34.3.1 uint32_t CCTrngState_t::LastTrngRosc

The last ROSC used for entropy collection

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

- cc_rnd_common_trng.h

2.6.35 CCTrngWorkBuff_t struct reference

```
#include <cc_rnd_common_trng.h>
```

2.6.35.1 Data Fields

- uint32_t **ccTrngIntWorkBuff** [**CC_TRNG_WORK_BUFFER_SIZE_WORDS**]

2.6.35.2 Detailed description

The definition of the RAM buffer, for internal use in instantiation or reseeding operations.

2.6.35.3 Field documentation

2.6.35.3.1 uint32_t CCTrngWorkBuff_t::ccTrngIntWorkBuff[CC_TRNG_WORK_BUFFER_SIZE_WORDS]

Internal buffer.

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

- cc_rnd_common_trng.h

2.6.36 EcdsaSignContext_t struct reference

```
#include <cc_ecpki_types.h>
```

2.6.36.1 Data Fields

- **CCEcpkiUserPrivKey_t ECDSA_SignerPrivKey**
- **CCHashUserContext_t hashUserCtxBuff**
- **CCHashResultBuf_t hashResult**
- uint32_t **hashResultSizeWords**
- **CCEcpkiHashOpMode_t hashMode**
- **CCEcdsaSignIntBuff_t ecdsaSignIntBuff**

2.6.36.2 Detailed description

The context definition for the signing operation.

2.6.36.3 Field documentation

2.6.36.3.1 CCEcpkiUserPrivKey_t **EcdsaSignContext_t::ECDSA_SignerPrivKey**

The data of the private key.

2.6.36.3.2 CCEcdsaSignIntBuff_t **EcdsaSignContext_t::ecdsaSignIntBuff**

Internal buffer.

2.6.36.3.3 CCEcpkiHashOpMode_t **EcdsaSignContext_t::hashMode**

The hash mode.

2.6.36.3.4 CCHashResultBuf_t **EcdsaSignContext_t::hashResult**

The hash result buffer.

2.6.36.3.5 uint32_t EcdsaSignContext_t::hashResultSizeWords

The size of the hash result in words.

2.6.36.3.6 CCHashUserContext_t **EcdsaSignContext_t::hashUserCtxBuff**

The hash context.

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

- **cc_ecpki_types.h**

2.6.37 EcdsaVerifyContext_t struct reference

```
#include <cc_ecpki_types_common.h>
```

2.6.37.1 Data Fields

- **CCEcpkiUserPublKey_t ECDSA_SignerPublKey**
- **CCHashUserContext_t hashUserCtxBuff**
- **CCHashResultBuf_t hashResult**
- uint32_t **hashResultSizeWords**
- **CCEcpkiHashOpMode_t hashMode**
- **CCEcdsaVerifyIntBuff_t ccEcdsaVerIntBuff**

2.6.37.2 Detailed description

The context definition for verification operation.

2.6.37.3 Field documentation

2.6.37.3.1 CCEcdsaVerifyIntBuff_t **EcdsaVerifyContext_t::ccEcdsaVerIntBuff**

Internal buffer.

2.6.37.3.2 CCEcpkiUserPublKey_t **EcdsaVerifyContext_t::ECDSA_SignerPublKey**

The data of the public key.

2.6.37.3.3 CCEcpkiHashOpMode_t **EcdsaVerifyContext_t::hashMode**

The hash mode.

2.6.37.3.4 CCHashResultBuf_t **EcdsaVerifyContext_t::hashResult**

The hash result.

2.6.37.3.5 uint32_t EcdsaVerifyContext_t::hashResultSizeWords

The size of the hash result in words.

2.6.37.3.6 CCHashUserContext_t **EcdsaVerifyContext_t::hashUserCtxBuff**

The hash context.

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

- **cc_ecpki_types_common.h**

2.6.38 File documentation

2.6.38.1 cc_aes_defs.h File Reference

This file contains the type definitions that are used by the CryptoCell AES APIs.

```
#include "cc_pal_types.h"

#include "cc_aes_defs_proj.h"
```

2.6.38.1.1 Data Structures

- struct **CCAesUserContext_t**
- struct **CCAesUserKeyData_t**
- struct **CCAesHwKeyData_t**

2.6.38.1.2 Macros

- #define **CC_AES_CRYPTO_BLOCK_SIZE_IN_WORDS** 4

- #define CC_AES_BLOCK_SIZE_IN_BYTES (CC_AES_CRYPTO_BLOCK_SIZE_IN_WORDS *sizeof(uint32_t))
- #define CC_AES_IV_SIZE_IN_WORDS CC_AES_CRYPTO_BLOCK_SIZE_IN_WORDS
- #define CC_AES_IV_SIZE_IN_BYTES (CC_AES_IV_SIZE_IN_WORDS *sizeof(uint32_t))

2.6.38.1.3 Typedefs

- typedef uint8_t CCAesIv_t[CC_AES_IV_SIZE_IN_BYTES]
- typedef uint8_t CCAesKeyBuffer_t[CC_AES_KEY_MAX_SIZE_IN_BYTES]
- typedef struct CCAesUserContext_t CCAesUserContext_t
- typedef struct CCAesUserKeyData_t CCAesUserKeyData_t
- typedef struct CCAesHwKeyData_t CCAesHwKeyData_t

2.6.38.1.4 Enumerations

- enum CCAesEncryptMode_t { CC_AES_ENCRYPT = 0, CC_AES_DECRYPT = 1, CC_AES_NUM_OF_ENCRYPT_MODES, CC_AES_ENCRYPT_MODE_LAST = 0x7FFFFFFF }
- enum CCAesOperationMode_t { CC_AES_MODE_ECB = 0, CC_AES_MODE_CBC = 1, CC_AES_MODE_CBC_MAC = 2, CC_AES_MODE_CTR = 3, CC_AES_MODE_XCBC_MAC = 4, CC_AES_MODE_CMAC = 5, CC_AES_MODE_XTS = 6, CC_AES_MODE_CBC_CTS = 7, CC_AES_MODE_OFB = 8, CC_AES_MODE_CFB = 9, CC_AES_NUM_OF_OPERATION_MODES, CC_AES_OPERATION_MODE_LAST = 0x7FFFFFFF }
- enum CCAesPaddingType_t { CC_AES_PADDING_NONE = 0, CC_AES_PADDING_PKCS7 = 1, CC_AES_NUM_OF_PADDING_TYPES, CC_AES_PADDING_TYPE_LAST = 0x7FFFFFFF }
- enum CCAesKeyType_t { CC_AES_USER_KEY = 0, CC_AES_PLATFORM_KEY = 1, CC_AES_CUSTOMER_KEY = 2, CC_AES_NUM_OF_KEY_TYPES, CC_AES_KEY_TYPE_LAST = 0x7FFFFFFF }

2.6.38.2 cc_aes_defs_proj.h File Reference

This file contains definitions that are used in the CryptoCell AES APIs.

```
#include "cc_pal_types.h"
```

2.6.38.2.1 Macros

- #define CC_AES_USER_CTX_SIZE_IN_WORDS 131
- #define CC_AES_KEY_MAX_SIZE_IN_WORDS 16
- #define CC_AES_KEY_MAX_SIZE_IN_BYTES (CC_AES_KEY_MAX_SIZE_IN_WORDS *sizeof(uint32_t))

2.6.38.3 cc_axi_ctrl.h File Reference

This file contains the AXI configuration control definitions.

```
#include "cc_pal_types.h"

#include "cc_error.h"
```

2.6.38.3.1 Data Structures

- union **CCAxiAceConst_t**
- union **CCAximCacheParams_t**
- struct **CCAxiFields_t**

2.6.38.3.2 Macros

- #define **CC_AXI_CTRL_ILEGALL_INPUT_ERROR** (**CC_AXI_CTRL_MODULE_ERROR_BASE** + 0x01)

2.6.38.3.3 Macro Definition Documentation

#define CC_AXI_CTRL_ILEGALL_INPUT_ERROR (**CC_AXI_CTRL_MODULE_ERROR_BASE** + 0x01)

This error is returned when one of the function inputs is illegal.

2.6.38.4 cc_bitops.h File Reference

This file defines bit-field operations macros.

2.6.38.4.1 Macros

- #define **CC_32BIT_MAX_VALUE** (0xFFFFFFFFUL)
- #define **BITMASK**(mask_size)
- #define **BITMASK_AT**(mask_size, mask_offset) (**BITMASK**(mask_size) << (mask_offset))
- #define **BITFIELD_GET**(word, bit_offset, bit_size) (((word) >> (bit_offset)) & **BITMASK**(bit_size))
- #define **BITFIELD_SET**(word, bit_offset, bit_size, new_val)
- #define **BITFIELD_U32_SHIFT_R**(res, val, shift)
- #define **BITFIELD_U32_SHIFT_L**(res, val, shift)
- #define **IS_ALIGNED**(val, align) (((uintptr_t)(val) & ((align) - 1)) == 0)
- #define **SWAP_ENDIAN**(word)
- #define **SWAP_TO_LE**(word) word
- #define **SWAP_TO_BE**(word) **SWAP_ENDIAN**(word)
- #define **ALIGN_TO_4BYTES**(x) (((unsigned long)(x) + (**CC_32BIT_WORD_SIZE**-1)) & ~(**CC_32BIT_WORD_SIZE**-1))
- #define **IS_MULT**(val, mult) (((val) & ((mult) - 1)) == 0)
- #define **IS_NULL_ADDR**(adr) (!(adr))

2.6.38.4.2 Macro Definition Documentation

#define ALIGN_TO_4BYTES(x) (((unsigned long)(x) + (**CC_32BIT_WORD_SIZE**-1)) & ~(**CC_32BIT_WORD_SIZE**-1))

Align X to uint32_t size.

#define BITFIELD_GET(word, bit_offset, bit_size) (((word) >> (bit_offset)) & **BITMASK**(bit_size))

Definition for getting bits value from a word.

#define BITFIELD_SET(word, bit_offset, bit_size, new_val)

```
do {     \
    word = ((word) & ~BITMASK_AT(bit_size, bit_offset)) |        \
        (((new_val) & BITMASK(bit_size)) << (bit_offset));   \
} while (0)
```

Definition for setting bits value from a word.

#define BITFIELD_U32_SHIFT_L(res, val, shift)

```
do { \
        if (((uint32_t)(shift)) < 32) { \
            (res) = (val) << (shift); \
        } else {\
            (res) = 0; \
        } \
    } while (0)
```

#define BITFIELD_U32_SHIFT_R(res, val, shift)

```
do { \
        if (((uint32_t)(shift)) < 32) { \
            (res) = (val) >> (shift); \
        } else {\
            (res) = 0; \
        } \
    } while (0)
```

#define BITMASK(mask_size)

```
(((mask_size) < 32) ?    \
    ((1UL << (mask_size)) - 1): 0xFFFFFFFFUL)
```

Definition for bitmask

#define BITMASK_AT(mask_size, mask_offset) (**BITMASK**(mask_size) << (mask_offset))

Definition for bitmask in a given offset.

#define CC_32BIT_MAX_VALUE (0xFFFFFFFFUL)

Definition of number of 32bit maximum value.

#define IS_ALIGNED(val, align) (((uintptr_t)(val) & ((align) - 1)) == 0)

Definition for is val aligned to "align" ("align" must be power of 2).

#define IS_MULT(val, mult) (((val) & ((mult) - 1)) == 0)

Definition for is val a multiple of "mult" ("mult" must be power of 2).

#define IS_NULL_ADDR(adr) (!(adr))

Definition for is NULL address.

#define SWAP_ENDIAN(word)

```
(((word) >> 24) | (((word) & 0x00FF0000) >> 8) | \

    (((word) & 0x0000FF00) << 8) | (((word) & 0x000000FF) << 24))
```

Definition swap endianity for 32 bits word.

#define SWAP_TO_BE(word) **SWAP_ENDIAN**(word)

Definition for swapping to BE.

#define SWAP_TO_LE(word) word

Definition for swapping to LE.

2.6.38.5 cc_cert_ctx.h File Reference

This file contains definitions that are required for CryptoCell's certification (FIPS or Chinese).

```
#include "cc_rsa_types.h"

#include "cc_ecpki_types.h"

#include "cc_dh.h"

#include "cc_rnd.h"

#include "cc_sm2.h"
```

2.6.38.5.1 Data Structures

- union **CCCertKatContext_t**

2.6.38.5.2 Macros

- #define **CCEcpkiKgCertContext_t CCSm2KeyGenCHCertContext_t**

2.6.38.6 cc_chinese_cert.h File Reference

This file contains definitions and APIs that are used in the CryptoCell Chinese Certification module.

```
#include "cc_pal_types.h"
```

2.6.38.6.1 Macros

- #define **CC_CH_CERT_STATE_NOT_SUPPORTED** 0x0
- #define **CC_CH_CERT_STATE_ERROR** 0x1
- #define **CC_CH_CERT_STATE_SUPPORTED** 0x2
- #define **CC_CH_CERT_STATE_CRYPTO_APPROVED** 0x4

- #define **CC_CH_CERT_CRYPTO_USAGE_SET_APPROVED**() **CC_ChCertCryptoUsageStateSet**(**CC_TEE_CH_CERT_CRYPTO_USAGE_STATE_APPROVED**)
- #define **CC_CH_CERT_CRYPTO_USAGE_SET_NON_APPROVED**() **CC_ChCertCryptoUsageStateSet**(**CC_TEE_CH_CERT_CRYPTO_USAGE_STATE_NON_APPROVED**)

2.6.38.6.2 Typedefs

- typedef uint32_t **CCChCertState_t**

2.6.38.6.3 Enumerations

- enum **CCChCertError_t** { **CC_TEE_CH_CERT_ERROR_OK** = 0, **CC_TEE_CH_CERT_ERROR_GENERAL**, **CC_TEE_CH_CERT_ERROR_SM4_ECB_PUT**, **CC_TEE_CH_CERT_ERROR_SM4_CBC_PUT**, **CC_TEE_CH_CERT_ERROR_SM4_CTR_PUT**, **CC_TEE_CH_CERT_ERROR_SM3_PUT**, **CC_TEE_CH_CERT_ERROR_SM2_SIGN_PUT**, **CC_TEE_CH_CERT_ERROR_SM2_KEY_GEN_COND**, **CC_TEE_CH_CERT_ERROR_RESERVE32B** = INT32_MAX }
- enum **CCChCertCryptoUsageState_t** { **CC_TEE_CH_CERT_CRYPTO_USAGE_STATE_NON_APPROVED** = 0, **CC_TEE_CH_CERT_CRYPTO_USAGE_STATE_APPROVED**, **CC_TEE_CH_CERT_CRYPTO_USAGE_STATE_RESERVE32B** = INT32_MAX }

2.6.38.6.4 Functions

- **CCError_t CC_ChCertErrorGet** (**CCChCertError_t** *pChCertError)

  This function is used to get the current Chinese certification error of the Arm CryptoCell TEE library.

- **CCError_t CC_ChCertStateGet** (**CCChCertState_t** *pChCertState)

  This function is used to get the current state of the Chinese certification state (Chinese certification state set to ON or OFF) and zeroization state of the Arm CryptoCell TEE library.

- **CCError_t CC_ChCertCryptoUsageStateSet** (**CCChCertCryptoUsageState_t** state)

  This function is used to set the permission (approved/non-approved) of the crypto operations in the suspended state of the Arm CryptoCell TEE library.

2.6.38.7 cc_chinese_cert_error.h File Reference

This file contains error codes definitions for CryptoCell Chinese certification module.

```
#include "cc_error.h"
```

2.6.38.7.1 Macros

- #define **CC_CH_CERT_ERROR** (**CC_CH_CERT_MODULE_ERROR_BASE** + 0x00UL)

2.6.38.8 cc_ecpki_build.h File Reference

This file defines functions for building key structures used in Elliptic Curves Cryptography (ECC).

```
#include "cc_error.h"

#include "cc_ecpki_types.h"
```

2.6.38.8.1 Macros

- #define **CC_EcpkiPubKeyBuild**(pDomain, pPubKeyIn, PublKeySizeInBytes, pUserPublKey) **CC_EcpkiPublKeyBuildAndCheck**((pDomain), (pPubKeyIn), (PublKeySizeInBytes), **CheckPointersAndSizesOnly**, (pUserPublKey), NULL)

  This macro calls **CC_EcpkiPublKeyBuildAndCheck()** function for building the public key while checking input pointers and sizes. For a description of the parameters see **CC_EcpkiPublKeyBuildAndCheck()**.

- #define **CC_EcpkiPubKeyBuildAndPartlyCheck**(pDomain, pPubKeyIn, PublKeySizeInBytes, pUserPublKey, pTempBuff) **CC_EcpkiPublKeyBuildAndCheck**((pDomain), (pPubKeyIn), (PublKeySizeInBytes), **ECpublKeyPartlyCheck**, (pUserPublKey), (pTempBuff))

  This macro calls CC_EcpkiPublKeyBuildAndCheck function for building the public key with partial validation of the key [SEC1] - 3.2.3. For a description of the parameters, see **CC_EcpkiPublKeyBuildAndCheck()**.

- #define **CC_EcpkiPubKeyBuildAndFullCheck**(pDomain, pPubKeyIn, PublKeySizeInBytes, pUserPublKey, pTempBuff) **CC_EcpkiPublKeyBuildAndCheck**((pDomain), (pPubKeyIn), (PublKeySizeInBytes), (**ECpublKeyFullCheck**), (pUserPublKey), (pTempBuff))

  This macro calls CC_EcpkiPublKeyBuildAndCheck function for building the public key with full validation of the key [SEC1] - 3.2.2. For a description of the parameters and return values, see **CC_EcpkiPublKeyBuildAndCheck()**.

2.6.38.8.2 Functions

- **CIMPORT_C CCError_t CC_EcpkiPrivKeyBuild** (const **CCEcpkiDomain_t** *pDomain, const uint8_t *pPrivKeyIn, size_t PrivKeySizeInBytes, **CCEcpkiUserPrivKey_t** *pUserPrivKey)

  Builds (imports) the user private key structure from an existing private key so that this structure can be used by other EC primitives. This function should be called before using of the private key. Input domain structure must be initialized by EC parameters and auxiliary values, using CC_EcpkiGetDomain() or **CC_EcpkiGetSm2Domain()** functions.

- **CIMPORT_C CCError_t CC_EcpkiPublKeyBuildAndCheck** (const **CCEcpkiDomain_t** *pDomain, uint8_t *pPubKeyIn, size_t PublKeySizeInBytes, **CCEcpkiUserPublKey_t** *pUserPublKey, **CCEcpkiBuildTempData_t** *pTempBuff)

  Builds a user public key structure from an imported public key, so it can be used by other EC primitives. When operating the EC cryptographic algorithms with imported EC public key, this function should be called before using of the public key.

- CIMPORT_C CCError_t CC_EcpkiPubKeyExport (CCEcpkiUserPublKey_t *pUserPublKey, CCEcpkiPointCompression_t compression, uint8_t *pExternPublKey, size_t *pPublKeySizeBytes)

  Converts an existing public key from internal representation to Big-Endian export representation. The function converts the X,Y coordinates of public key EC point to big endianness, and sets the public key.

2.6.38.9 cc_ecpki_domain_sm2.h File Reference

This file defines the SM2 get domain API.

```
#include "cc_pal_types.h"
```

```
#include "cc_ecpki_types.h"
```

2.6.38.9.1 Functions

- const CCEcpkiDomain_t *CC_EcpkiGetSm2Domain (void)

  The function returns the domain pointer of SM2.

2.6.38.10 cc_ecpki_error.h File Reference

This file contains the definitions of the CryptoCell ECPKI errors.

```
#include "cc_error.h"
```

2.6.38.10.1 Macros

- #define CC_ECPKI_ILLEGAL_DOMAIN_ID_ERROR (CC_ECPKI_MODULE_ERROR_BASE + 0x1UL)
- #define CC_ECPKI_DOMAIN_PTR_ERROR (CC_ECPKI_MODULE_ERROR_BASE + 0x2UL)
- #define CC_ECPKI_GEN_KEY_INVALID_PRIVATE_KEY_PTR_ERROR (CC_ECPKI_MODULE_ERROR_BASE + 0x3UL)
- #define CC_ECPKI_GEN_KEY_INVALID_PUBLIC_KEY_PTR_ERROR (CC_ECPKI_MODULE_ERROR_BASE + 0x4UL)
- #define CC_ECPKI_GEN_KEY_INVALID_TEMP_DATA_PTR_ERROR (CC_ECPKI_MODULE_ERROR_BASE + 0x5UL)
- #define CC_ECPKI_RND_CONTEXT_PTR_ERROR (CC_ECPKI_MODULE_ERROR_BASE + 0x6UL)
- #define CC_ECPKI_BUILD_KEY_INVALID_COMPRESSION_MODE_ERROR (CC_ECPKI_MODULE_ERROR_BASE + 0x07UL)
- #define CC_ECPKI_BUILD_KEY_ILLEGAL_DOMAIN_ID_ERROR (CC_ECPKI_MODULE_ERROR_BASE + 0x08UL)

- #define CC_ECPKI_BUILD_KEY_INVALID_PRIV_KEY_IN_PTR_ERROR (CC_ECPKI_MODULE_ERROR_BASE + 0x09UL)
- #define CC_ECPKI_BUILD_KEY_INVALID_USER_PRIV_KEY_PTR_ERROR (CC_ECPKI_MODULE_ERROR_BASE + 0x0AUL)
- #define CC_ECPKI_BUILD_KEY_INVALID_PRIV_KEY_SIZE_ERROR (CC_ECPKI_MODULE_ERROR_BASE + 0x0BUL)
- #define CC_ECPKI_BUILD_KEY_INVALID_PRIV_KEY_DATA_ERROR (CC_ECPKI_MODULE_ERROR_BASE + 0x0CUL)
- #define CC_ECPKI_BUILD_KEY_INVALID_PUBL_KEY_IN_PTR_ERROR (CC_ECPKI_MODULE_ERROR_BASE + 0x0DUL)
- #define CC_ECPKI_BUILD_KEY_INVALID_USER_PUBL_KEY_PTR_ERROR (CC_ECPKI_MODULE_ERROR_BASE + 0x0EUL)
- #define CC_ECPKI_BUILD_KEY_INVALID_PUBL_KEY_SIZE_ERROR (CC_ECPKI_MODULE_ERROR_BASE + 0x0FUL)
- #define CC_ECPKI_BUILD_KEY_INVALID_PUBL_KEY_DATA_ERROR (CC_ECPKI_MODULE_ERROR_BASE + 0x10UL)
- #define CC_ECPKI_BUILD_KEY_INVALID_CHECK_MODE_ERROR (CC_ECPKI_MODULE_ERROR_BASE + 0x11UL)
- #define CC_ECPKI_BUILD_KEY_INVALID_TEMP_BUFF_PTR_ERROR (CC_ECPKI_MODULE_ERROR_BASE + 0x12UL)
- #define CC_ECPKI_EXPORT_PUBL_KEY_INVALID_USER_PUBL_KEY_PTR_ERROR (CC_ECPKI_MODULE_ERROR_BASE + 0x14UL)
- #define CC_ECPKI_EXPORT_PUBL_KEY_ILLEGAL_COMPRESSION_MODE_ERROR (CC_ECPKI_MODULE_ERROR_BASE + 0x15UL)
- #define CC_ECPKI_EXPORT_PUBL_KEY_INVALID_EXTERN_PUBL_KEY_PTR_ERROR (CC_ECPKI_MODULE_ERROR_BASE + 0x16UL)
- #define CC_ECPKI_EXPORT_PUBL_KEY_INVALID_PUBL_KEY_SIZE_PTR_ERROR (CC_ECPKI_MODULE_ERROR_BASE + 0x17UL)
- #define CC_ECPKI_EXPORT_PUBL_KEY_INVALID_PUBL_KEY_SIZE_ERROR (CC_ECPKI_MODULE_ERROR_BASE + 0x18UL)
- #define CC_ECPKI_EXPORT_PUBL_KEY_ILLEGAL_DOMAIN_ID_ERROR (CC_ECPKI_MODULE_ERROR_BASE + 0x19UL)
- #define CC_ECPKI_EXPORT_PUBL_KEY_ILLEGAL_VALIDATION_TAG_ERROR (CC_ECPKI_MODULE_ERROR_BASE + 0x1AUL)
- #define CC_ECPKI_EXPORT_PUBL_KEY_INVALID_PUBL_KEY_DATA_ERROR (CC_ECPKI_MODULE_ERROR_BASE + 0x1BUL)
- #define CC_ECPKI_BUILD_DOMAIN_ID_IS_NOT_VALID_ERROR (CC_ECPKI_MODULE_ERROR_BASE + 0x20UL)
- #define CC_ECPKI_BUILD_DOMAIN_DOMAIN_PTR_ERROR (CC_ECPKI_MODULE_ERROR_BASE + 0x21UL)

- #define CC_ECPKI_BUILD_DOMAIN_EC_PARAMETR_PTR_ERROR (CC_ECPKI_MODULE_ERROR_BASE + 0x22UL)
- #define CC_ECPKI_BUILD_DOMAIN_EC_PARAMETR_SIZE_ERROR (CC_ECPKI_MODULE_ERROR_BASE + 0x23UL)
- #define CC_ECPKI_BUILD_DOMAIN_COFACTOR_PARAMS_ERROR (CC_ECPKI_MODULE_ERROR_BASE + 0x24UL)
- #define CC_ECPKI_BUILD_DOMAIN_SECURITY_STRENGTH_ERROR (CC_ECPKI_MODULE_ERROR_BASE + 0x25UL)
- #define CC_ECPKI_BUILD_SCA_RESIST_ILLEGAL_MODE_ERROR (CC_ECPKI_MODULE_ERROR_BASE + 0x26UL)
- #define CC_ECPKI_INTERNAL_ERROR (CC_ECPKI_MODULE_ERROR_BASE + 0x30UL)
- #define CC_ECDH_SVDP_DH_INVALID_PARTNER_PUBL_KEY_PTR_ERROR (CC_ECPKI_MODULE_ERROR_BASE + 0x31UL)
- #define CC_ECDH_SVDP_DH_PARTNER_PUBL_KEY_VALID_TAG_ERROR (CC_ECPKI_MODULE_ERROR_BASE + 0x32UL)
- #define CC_ECDH_SVDP_DH_INVALID_USER_PRIV_KEY_PTR_ERROR (CC_ECPKI_MODULE_ERROR_BASE + 0x33UL)
- #define CC_ECDH_SVDP_DH_USER_PRIV_KEY_VALID_TAG_ERROR (CC_ECPKI_MODULE_ERROR_BASE + 0x34UL)
- #define CC_ECDH_SVDP_DH_INVALID_SHARED_SECRET_VALUE_PTR_ERROR (CC_ECPKI_MODULE_ERROR_BASE + 0x35UL)
- #define CC_ECDH_SVDP_DH_INVALID_TEMP_DATA_PTR_ERROR (CC_ECPKI_MODULE_ERROR_BASE + 0x36UL)
- #define CC_ECDH_SVDP_DH_INVALID_SHARED_SECRET_VALUE_SIZE_PTR_ERROR (CC_ECPKI_MODULE_ERROR_BASE + 0x37UL)
- #define CC_ECDH_SVDP_DH_INVALID_SHARED_SECRET_VALUE_SIZE_ERROR (CC_ECPKI_MODULE_ERROR_BASE + 0x38UL)
- #define CC_ECDH_SVDP_DH_ILLEGAL_DOMAIN_ID_ERROR (CC_ECPKI_MODULE_ERROR_BASE + 0x39UL)
- #define CC_ECDH_SVDP_DH_NOT_CONCENT_PUBL_AND_PRIV_DOMAIN_ID_ERROR (CC_ECPKI_MODULE_ERROR_BASE + 0x3AUL)
- #define CC_ECDSA_SIGN_INVALID_DOMAIN_ID_ERROR (CC_ECPKI_MODULE_ERROR_BASE + 0x50UL)
- #define CC_ECDSA_SIGN_INVALID_USER_CONTEXT_PTR_ERROR (CC_ECPKI_MODULE_ERROR_BASE + 0x51UL)
- #define CC_ECDSA_SIGN_INVALID_USER_PRIV_KEY_PTR_ERROR (CC_ECPKI_MODULE_ERROR_BASE + 0x52UL)
- #define CC_ECDSA_SIGN_ILLEGAL_HASH_OP_MODE_ERROR (CC_ECPKI_MODULE_ERROR_BASE + 0x53UL)
- #define CC_ECDSA_SIGN_INVALID_MESSAGE_DATA_IN_PTR_ERROR (CC_ECPKI_MODULE_ERROR_BASE + 0x54UL)

- #define CC_ECDSA_SIGN_INVALID_MESSAGE_DATA_IN_SIZE_ERROR (CC_ECPKI_MODULE_ERROR_BASE + 0x55UL)
- #define CC_ECDSA_SIGN_USER_CONTEXT_VALIDATION_TAG_ERROR (CC_ECPKI_MODULE_ERROR_BASE + 0x57UL)
- #define CC_ECDSA_SIGN_USER_PRIV_KEY_VALIDATION_TAG_ERROR (CC_ECPKI_MODULE_ERROR_BASE + 0x58UL)
- #define CC_ECDSA_SIGN_INVALID_SIGNATURE_OUT_PTR_ERROR (CC_ECPKI_MODULE_ERROR_BASE + 0x60UL)
- #define CC_ECDSA_SIGN_INVALID_SIGNATURE_OUT_SIZE_PTR_ERROR (CC_ECPKI_MODULE_ERROR_BASE + 0x61UL)
- #define CC_ECDSA_SIGN_INVALID_SIGNATURE_OUT_SIZE_ERROR (CC_ECPKI_MODULE_ERROR_BASE + 0x62UL)
- #define CC_ECDSA_SIGN_INVALID_IS_EPHEMER_KEY_INTERNAL_ERROR (CC_ECPKI_MODULE_ERROR_BASE + 0x63UL)
- #define CC_ECDSA_SIGN_INVALID_EPHEMERAL_KEY_PTR_ERROR (CC_ECPKI_MODULE_ERROR_BASE + 0x64UL)
- #define CC_ECDSA_SIGN_INVALID_RND_CONTEXT_PTR_ERROR (CC_ECPKI_MODULE_ERROR_BASE + 0x65UL)
- #define CC_ECDSA_SIGN_INVALID_RND_FUNCTION_PTR_ERROR (CC_ECPKI_MODULE_ERROR_BASE + 0x66UL)
- #define CC_ECDSA_SIGN_SIGNING_ERROR (CC_ECPKI_MODULE_ERROR_BASE + 0x67UL)
- #define CC_ECDSA_VERIFY_INVALID_DOMAIN_ID_ERROR (CC_ECPKI_MODULE_ERROR_BASE + 0x70UL)
- #define CC_ECDSA_VERIFY_INVALID_USER_CONTEXT_PTR_ERROR (CC_ECPKI_MODULE_ERROR_BASE + 0x71UL)
- #define CC_ECDSA_VERIFY_INVALID_SIGNER_PUBL_KEY_PTR_ERROR (CC_ECPKI_MODULE_ERROR_BASE + 0x72UL)
- #define CC_ECDSA_VERIFY_ILLEGAL_HASH_OP_MODE_ERROR (CC_ECPKI_MODULE_ERROR_BASE + 0x73UL)
- #define CC_ECDSA_VERIFY_INVALID_SIGNATURE_IN_PTR_ERROR (CC_ECPKI_MODULE_ERROR_BASE + 0x76UL)
- #define CC_ECDSA_VERIFY_INVALID_SIGNATURE_SIZE_ERROR (CC_ECPKI_MODULE_ERROR_BASE + 0x77UL)
- #define CC_ECDSA_VERIFY_INVALID_MESSAGE_DATA_IN_PTR_ERROR (CC_ECPKI_MODULE_ERROR_BASE + 0x80UL)
- #define CC_ECDSA_VERIFY_INVALID_MESSAGE_DATA_IN_SIZE_ERROR (CC_ECPKI_MODULE_ERROR_BASE + 0x81UL)
- #define CC_ECDSA_VERIFY_USER_CONTEXT_VALIDATION_TAG_ERROR (CC_ECPKI_MODULE_ERROR_BASE + 0x82UL)

- #define **CC_ECDSA_VERIFY_SIGNER_PUBL_KEY_VALIDATION_TAG_ERROR** (**CC_ECPKI_MODULE_ERROR_BASE** + 0x83UL)
- #define **CC_ECDSA_VERIFY_INCONSISTENT_VERIFY_ERROR** (**CC_ECPKI_MODULE_ERROR_BASE** + 0x84UL)
- #define **CC_ECC_ILLEGAL_HASH_MODE_ERROR** (**CC_ECPKI_MODULE_ERROR_BASE** + 0x85UL)
- #define **CC_ECPKI_INVALID_RND_FUNC_PTR_ERROR** (**CC_ECPKI_MODULE_ERROR_BASE** + 0x90UL)
- #define **CC_ECPKI_INVALID_RND_CTX_PTR_ERROR** (**CC_ECPKI_MODULE_ERROR_BASE** + 0x91UL)
- #define **CC_ECPKI_INVALID_DOMAIN_ID_ERROR** (**CC_ECPKI_MODULE_ERROR_BASE** + 0x92UL)
- #define **CC_ECPKI_INVALID_PRIV_KEY_TAG_ERROR** (**CC_ECPKI_MODULE_ERROR_BASE** + 0x93UL)
- #define **CC_ECPKI_INVALID_PUBL_KEY_TAG_ERROR** (**CC_ECPKI_MODULE_ERROR_BASE** + 0x94UL)
- #define **CC_ECPKI_INVALID_DATA_IN_PASSED_STRUCT_ERROR** (**CC_ECPKI_MODULE_ERROR_BASE** + 0x95UL)
- #define **CC_ECPKI_INVALID_BASE_POINT_PTR_ERROR** (**CC_ECPKI_MODULE_ERROR_BASE** + 0x96UL)
- #define **CC_ECPKI_INVALID_OUT_HASH_PTR_ERROR** (**CC_ECPKI_MODULE_ERROR_BASE** + 0x97UL)
- #define **CC_ECPKI_INVALID_OUT_HASH_SIZE_ERROR** (**CC_ECPKI_MODULE_ERROR_BASE** + 0x98UL)
- #define **CC_ECPKI_INVALID_IN_HASH_PTR_ERROR** (**CC_ECPKI_MODULE_ERROR_BASE** + 0x99UL)
- #define **CC_ECPKI_INVALID_IN_HASH_SIZE_ERROR** (**CC_ECPKI_MODULE_ERROR_BASE** + 0x9AUL)
- #define **CC_ECPKI_SM2_INVALID_KE_CONTEXT_PTR** (**CC_ECPKI_MODULE_ERROR_BASE** + 0xA0UL)
- #define **CC_ECPKI_SM2_INVALID_ID_PTR** (**CC_ECPKI_MODULE_ERROR_BASE** + 0xA1UL)
- #define **CC_ECPKI_SM2_INVALID_ID_SIZE** (**CC_ECPKI_MODULE_ERROR_BASE** + 0xA2UL)
- #define **CC_ECPKI_SM2_INVALID_IN_PARAM_SIZE** (**CC_ECPKI_MODULE_ERROR_BASE** + 0xA3UL)
- #define **CC_ECPKI_SM2_INVALID_OUT_PARAM_SIZE** (**CC_ECPKI_MODULE_ERROR_BASE** + 0xA4UL)
- #define **CC_ECPKI_SM2_INVALID_OUT_PARAM_PTR** (**CC_ECPKI_MODULE_ERROR_BASE** + 0xA5UL)
- #define **CC_ECPKI_SM2_INVALID_CONTEXT** (**CC_ECPKI_MODULE_ERROR_BASE** + 0xA6UL)

- #define CC_ECPKI_SM2_INVALID_EPHEMERAL_PUB_IN_PTR (CC_ECPKI_MODULE_ERROR_BASE + 0xA7UL)
- #define CC_ECPKI_SM2_INVALID_EPHEMERAL_PUB_OUT_PTR (CC_ECPKI_MODULE_ERROR_BASE + 0xA8UL)
- #define CC_ECPKI_SM2_INVALID_SHARED_SECRET_OUT_PTR (CC_ECPKI_MODULE_ERROR_BASE + 0xA9UL)
- #define CC_ECPKI_SM2_INVALID_SHARED_SECRET_IN_PTR (CC_ECPKI_MODULE_ERROR_BASE + 0xAAUL)
- #define CC_ECPKI_SM2_INVALID_IN_PARAM_PTR (CC_ECPKI_MODULE_ERROR_BASE + 0xABUL)
- #define CC_ECPKI_SM2_INVALID_EPHEMERAL_PRIV_IN_PTR (CC_ECPKI_MODULE_ERROR_BASE + 0xACUL)
- #define CC_ECPKI_SM2_CONFIRMATION_FAILED (CC_ECPKI_MODULE_ERROR_BASE + 0xADUL)
- #define CC_ECIES_INVALID_PUBL_KEY_PTR_ERROR (CC_ECPKI_MODULE_ERROR_BASE + 0xE0UL)
- #define CC_ECIES_INVALID_PUBL_KEY_TAG_ERROR (CC_ECPKI_MODULE_ERROR_BASE + 0xE1UL)
- #define CC_ECIES_INVALID_PRIV_KEY_PTR_ERROR (CC_ECPKI_MODULE_ERROR_BASE + 0xE2UL)
- #define CC_ECIES_INVALID_PRIV_KEY_TAG_ERROR (CC_ECPKI_MODULE_ERROR_BASE + 0xE3UL)
- #define CC_ECIES_INVALID_PRIV_KEY_VALUE_ERROR (CC_ECPKI_MODULE_ERROR_BASE + 0xE4UL)
- #define CC_ECIES_INVALID_KDF_DERIV_MODE_ERROR (CC_ECPKI_MODULE_ERROR_BASE + 0xE5UL)
- #define CC_ECIES_INVALID_KDF_HASH_MODE_ERROR (CC_ECPKI_MODULE_ERROR_BASE + 0xE6UL)
- #define CC_ECIES_INVALID_SECRET_KEY_PTR_ERROR (CC_ECPKI_MODULE_ERROR_BASE + 0xE7UL)
- #define CC_ECIES_INVALID_SECRET_KEY_SIZE_ERROR (CC_ECPKI_MODULE_ERROR_BASE + 0xE8UL)
- #define CC_ECIES_INVALID_CIPHER_DATA_PTR_ERROR (CC_ECPKI_MODULE_ERROR_BASE + 0xE9UL)
- #define CC_ECIES_INVALID_CIPHER_DATA_SIZE_PTR_ERROR (CC_ECPKI_MODULE_ERROR_BASE + 0xEAUL)
- #define CC_ECIES_INVALID_CIPHER_DATA_SIZE_ERROR (CC_ECPKI_MODULE_ERROR_BASE + 0xEBUL)
- #define CC_ECIES_INVALID_TEMP_DATA_PTR_ERROR (CC_ECPKI_MODULE_ERROR_BASE + 0xECUL)

- #define CC_ECIES_INVALID_TEMP_DATA_SIZE_ERROR (CC_ECPKI_MODULE_ERROR_BASE + 0xEDUL)
- #define CC_ECIES_INVALID_EPHEM_KEY_PAIR_PTR_ERROR (CC_ECPKI_MODULE_ERROR_BASE + 0xEEUL)
- #define CC_ECIES_INVALID_PTR (CC_ECPKI_MODULE_ERROR_BASE + 0xEFUL)

2.6.38.11 cc_ecpki_kg.h File Reference

This file defines the API for generation of ECC private and public keys.

```
#include "cc_error.h"

#include "cc_rnd_common.h"

#include "cc_ecpki_types.h"

#include "cc_cert_ctx.h"
```

2.6.38.11.1 Functions

- **CIMPORT_C CCError_t CC_EcpkiKeyPairGenerate** (**CCRndGenerateVectWorkFunc_t** f_rng, void *p_rng, const **CCEcpkiDomain_t** *pDomain, **CCEcpkiUserPrivKey_t** *pUserPrivKey, **CCEcpkiUserPublKey_t** *pUserPublKey, **CCEcpkiKgTempData_t** *pTempData, **CCEcpkiKgCertContext_t** *pFipsCtx)

  Generates a pair of private and public keys in internal representation according to ANSI X9.62-2005: Public Key Cryptography for the Financial Services Industry, The Elliptic Curve Digital Signature Algorithm (ECDSA) standard.

- **CIMPORT_C CCError_t CC_EcpkiKeyPairGenerateBase** (**CCRndGenerateVectWorkFunc_t** f_rng, void *p_rng, const **CCEcpkiDomain_t** *pDomain, const uint32_t *ecX_ptr, const uint32_t *ecY_ptr, **CCEcpkiUserPrivKey_t** *pUserPrivKey, **CCEcpkiUserPublKey_t** *pUserPublKey, **CCEcpkiKgTempData_t** *pTempData, **CCEcpkiKgCertContext_t** *pFipsCtx)

  Generates a pair of private and public keys using a configurable base point in internal representation according to ANSI X9.62-2005: Public Key Cryptography for the Financial Services Industry, The Elliptic Curve Digital Signature Algorithm (ECDSA) standard.

2.6.38.12 cc_ecpki_types.h File Reference

This file contains all the type definitions that are used for the CryptoCell ECPKI APIs.

```
#include "cc_bitops.h"

#include "cc_pal_types_plat.h"

#include "cc_hash_defs.h"

#include "cc_pka_defs_hw.h"
```

```
#include "cc_pal_compiler.h"

#include "cc_ecpki_types_common.h"
```

2.6.38.12.1 Data Structures

- struct CCEcpkiPointAffine_t
- struct EcdsaSignContext_t
- struct CCEcdsaSignUserContext_t

  The context definition of the user for the signing operation.
- struct CCEcdsaFipsKatContext_t
- struct CCEcdhFipsKatContext_t
- struct CCEcpkiKgFipsContext_t

2.6.38.12.2 Macros

- #define CC_ECPKI_FIPS_ORDER_LENGTH (256/CC_BITS_IN_BYTE)

2.6.38.12.3 Typedefs

- typedef uint32_t CCEcdsaSignIntBuff_t[CC_PKA_ECDSA_SIGN_BUFF_MAX_LENGTH_IN_WORDS]
- typedef struct CCEcdsaSignUserContext_t CCEcdsaSignUserContext_t

  The context definition of the user for the signing operation.
- typedef struct CCEcdsaFipsKatContext_t CCEcdsaFipsKatContext_t
- typedef struct CCEcdhFipsKatContext_t CCEcdhFipsKatContext_t
- typedef struct CCEcpkiKgFipsContext_t CCEcpkiKgFipsContext_t

2.6.38.13 cc_ecpki_types_common.h File Reference

This file contains all the type definitions that are used for the CryptoCell ECPKI APIs.

```
#include "cc_pal_types_plat.h"

#include "cc_hash_defs.h"

#include "cc_pka_defs_hw.h"
```

2.6.38.13.1 Data Structures

- struct CCEcpkiDomain_t

  The structure containing the EC domain parameters in little-endian form.
- struct CCEcpkiPublKey_t

- struct **CCEcpkiUserPublKey_t**

  The user structure prototype of the EC public key.

- struct **CCEcpkiPrivKey_t**
- struct **CCEcpkiUserPrivKey_t**

  The user structure prototype of the EC private key.

- struct **CCEcdhTempData_t**
- struct **CCEcpkiBuildTempData_t**
- struct **EcdsaVerifyContext_t**
- struct **CCEcdsaVerifyUserContext_t**

  The context definition of the user for the verification operation.

- struct **CCEcpkiKgTempData_t**
- struct **CCEciesTempData_t**

2.6.38.13.2 Macros

- #define **CC_PKA_DOMAIN_LLF_BUFF_SIZE_IN_WORDS** (10 + 3***CC_ECPKI_MODUL_MAX_LENGTH_IN_WORDS**)

2.6.38.13.3 Typedefs

- typedef struct **CCEcpkiUserPublKey_t CCEcpkiUserPublKey_t**

  The user structure prototype of the EC public key.

- typedef struct **CCEcpkiUserPrivKey_t CCEcpkiUserPrivKey_t**

  The user structure prototype of the EC private key.

- typedef struct **CCEcdhTempData_t CCEcdhTempData_t**
- typedef struct **CCEcpkiBuildTempData_t CCEcpkiBuildTempData_t**
- typedef uint32_t **CCEcdsaVerifyIntBuff_t**[**CC_PKA_ECDSA_VERIFY_BUFF_MAX_LENGTH_IN_WORDS**]
- typedef struct **CCEcdsaVerifyUserContext_t CCEcdsaVerifyUserContext_t**

  The context definition of the user for the verification operation.

- typedef struct **CCEcpkiKgTempData_t CCEcpkiKgTempData_t**
- typedef struct **CCEciesTempData_t CCEciesTempData_t**

2.6.38.13.4 Enumerations

- enum **CCEcpkiDomainID_t** { **CC_ECPKI_DomainID_secp192k1**, **CC_ECPKI_DomainID_secp192r1**, **CC_ECPKI_DomainID_secp224k1**, **CC_ECPKI_DomainID_secp224r1**, **CC_ECPKI_DomainID_secp256k1**, **CC_ECPKI_DomainID_secp256r1**, **CC_ECPKI_DomainID_secp384r1**, **CC_ECPKI_DomainID_secp521r1**, **CC_ECPKI_DomainID_bp256r1**,

CC_ECPKI_DomainID_Builded, **CC_ECPKI_DomainID_sm2**, **CC_ECPKI_DomainID_OffMode**, **CC_ECPKI_DomainIDLast** = 0x7FFFFFFF }

EC domain identifiers.

- enum **CCEcpkiHashOpMode_t** { **CC_ECPKI_HASH_SHA1_mode** = 0, **CC_ECPKI_HASH_SHA224_mode** = 1, **CC_ECPKI_HASH_SHA256_mode** = 2, **CC_ECPKI_HASH_SHA384_mode** = 3, **CC_ECPKI_HASH_SHA512_mode** = 4, **CC_ECPKI_AFTER_HASH_SHA1_mode** = 5, **CC_ECPKI_AFTER_HASH_SHA224_mode** = 6, **CC_ECPKI_AFTER_HASH_SHA256_mode** = 7, **CC_ECPKI_AFTER_HASH_SHA384_mode** = 8, **CC_ECPKI_AFTER_HASH_SHA512_mode** = 9, **CC_ECPKI_HASH_NumOfModes**, **CC_ECPKI_HASH_OpModeLast** = 0x7FFFFFFF }

  Hash operation mode.
- enum **CCEcpkiPointCompression_t** { **CC_EC_PointCompressed** = 2, **CC_EC_PointUncompressed** = 4, **CC_EC_PointContWrong** = 5, **CC_EC_PointHybrid** = 6, **CC_EC_PointCompresOffMode** = 8, **CC_ECPKI_PointCompressionLast** = 0x7FFFFFFF }
- enum **ECPublKeyCheckMode_t** { **CheckPointersAndSizesOnly** = 0, **ECpublKeyPartlyCheck** = 1, **ECpublKeyFullCheck** = 2, **PublKeyChecingOffMode**, **EC_PublKeyCheckModeLast** = 0x7FFFFFFF }
- enum **CCEcpkiScaProtection_t** { **SCAP_Inactive**, **SCAP_Active**, **SCAP_OFF_MODE**, **SCAP_LAST** = 0x7FFFFFFF }

2.6.38.14 cc_error.h File Reference

This file defines the error return code types and the numbering spaces for each module of the layers listed.

```
#include "cc_pal_types.h"
```

2.6.38.14.1 Macros

- #define **CC_ERROR_BASE** 0x00F00000UL
- #define **CC_ERROR_LAYER_RANGE** 0x00010000UL
- #define **CC_ERROR_MODULE_RANGE** 0x00000100UL
- #define **CC_LAYER_ERROR_IDX** 0x00UL
- #define **LLF_LAYER_ERROR_IDX** 0x01UL
- #define **GENERIC_ERROR_IDX** 0x05UL
- #define **AES_ERROR_IDX** 0x00UL
- #define **DES_ERROR_IDX** 0x01UL
- #define **HASH_ERROR_IDX** 0x02UL
- #define **HMAC_ERROR_IDX** 0x03UL
- #define **RSA_ERROR_IDX** 0x04UL

- #define **DH_ERROR_IDX** 0x05UL
- #define **ECPKI_ERROR_IDX** 0x08UL
- #define **RND_ERROR_IDX** 0x0CUL
- #define **COMMON_ERROR_IDX** 0x0DUL
- #define **KDF_ERROR_IDX** 0x11UL
- #define **HKDF_ERROR_IDX** 0x12UL
- #define **AESCCM_ERROR_IDX** 0x15UL
- #define **FIPS_ERROR_IDX** 0x17UL
- #define **CH_CERT_ERROR_IDX** 0x18UL
- #define **PKA_MODULE_ERROR_IDX** 0x21UL
- #define **CHACHA_ERROR_IDX** 0x22UL
- #define **EC_MONT_EDW_ERROR_IDX** 0x23UL
- #define **CHACHA_POLY_ERROR_IDX** 0x24UL
- #define **POLY_ERROR_IDX** 0x25UL
- #define **SRP_ERROR_IDX** 0x26UL
- #define **AESGCM_ERROR_IDX** 0x27UL
- #define **AES_KEYWRAP_ERROR_IDX** 0x28UL
- #define **MNG_ERROR_IDX** 0x29UL
- #define **PROD_ERROR_IDX** 0x2AUL
- #define **FFCDH_ERROR_IDX** 0x2BUL
- #define **FFC_DOMAIN_ERROR_IDX** 0x2CUL
- #define **SB_ECC_ERROR_IDX_** 0x2DUL
- #define **EXT_DMA_ERROR_IDX** 0x2EUL
- #define **TRNG_ERROR_IDX** 0x2FUL
- #define **SM3_ERROR_IDX** 0x30UL
- #define **SM4_ERROR_IDX** 0x31UL
- #define **CPP_ERROR_IDX** 0x32UL
- #define **AXI_CTRL_ERROR_IDX** 0x33UL
- #define **CC_AES_MODULE_ERROR_BASE**
- #define **CC_DES_MODULE_ERROR_BASE**
- #define **CC_HASH_MODULE_ERROR_BASE**
- #define **CC_HMAC_MODULE_ERROR_BASE**
- #define **CC_RSA_MODULE_ERROR_BASE**
- #define **CC_DH_MODULE_ERROR_BASE**

- #define CC_ECPKI_MODULE_ERROR_BASE
- #define LLF_ECPKI_MODULE_ERROR_BASE
- #define CC_RND_MODULE_ERROR_BASE
- #define LLF_RND_MODULE_ERROR_BASE
- #define CC_COMMON_MODULE_ERROR_BASE
- #define CC_KDF_MODULE_ERROR_BASE
- #define CC_HKDF_MODULE_ERROR_BASE
- #define CC_AESCCM_MODULE_ERROR_BASE
- #define CC_FIPS_MODULE_ERROR_BASE
- #define CC_CH_CERT_MODULE_ERROR_BASE
- #define PKA_MODULE_ERROR_BASE
- #define CC_CHACHA_MODULE_ERROR_BASE
- #define CC_EC_MONT_EDW_MODULE_ERROR_BASE
- #define CC_CHACHA_POLY_MODULE_ERROR_BASE
- #define CC_POLY_MODULE_ERROR_BASE
- #define CC_SRP_MODULE_ERROR_BASE
- #define CC_AESGCM_MODULE_ERROR_BASE
- #define CC_AES_KEYWRAP_MODULE_ERROR_BASE
- #define CC_MNG_MODULE_ERROR_BASE
- #define CC_PROD_MODULE_ERROR_BASE
- #define CC_FFCDH_MODULE_ERROR_BASE
- #define CC_FFC_DOMAIN_MODULE_ERROR_BASE
- #define CC_EXT_DMA_MODULE_ERROR_BASE
- #define CC_TRNG_MODULE_ERROR_BASE
- #define CC_SM3_MODULE_ERROR_BASE
- #define CC_SM4_MODULE_ERROR_BASE
- #define CC_CPP_MODULE_ERROR_BASE
- #define CC_AXI_CTRL_MODULE_ERROR_BASE
- #define GENERIC_ERROR_BASE (CC_ERROR_BASE + (CC_ERROR_LAYER_RANGE *GENERIC_ERROR_IDX))
- #define CC_FATAL_ERROR (GENERIC_ERROR_BASE + 0x00UL)
- #define CC_OUT_OF_RESOURCE_ERROR (GENERIC_ERROR_BASE + 0x01UL)
- #define CC_ILLEGAL_RESOURCE_VAL_ERROR (GENERIC_ERROR_BASE + 0x02UL)
- #define CC_CRYPTO_RETURN_ERROR(retCode, retcodeInfo, funcHandler) ((retCode) == 0 ? CC_OK: funcHandler(retCode, retcodeInfo))

2.6.38.15 cc_hash_defs.h File Reference

This file contains definitions of the CryptoCell hash APIs.

```
#include "cc_pal_types.h"
```

```
#include "cc_error.h"
```

```
#include "cc_hash_defs_proj.h"
```

2.6.38.15.1 Data Structures

- struct **CCHashUserContext_t**

2.6.38.15.2 Macros

- #define **CC_HASH_RESULT_SIZE_IN_WORDS** 16
- #define **CC_HASH_MD5_DIGEST_SIZE_IN_BYTES** 16
- #define **CC_HASH_MD5_DIGEST_SIZE_IN_WORDS** 4
- #define **CC_HASH_SHA1_DIGEST_SIZE_IN_BYTES** 20
- #define **CC_HASH_SHA1_DIGEST_SIZE_IN_WORDS** 5
- #define **CC_HASH_SHA224_DIGEST_SIZE_IN_WORDS** 7
- #define **CC_HASH_SHA256_DIGEST_SIZE_IN_WORDS** 8
- #define **CC_HASH_SHA384_DIGEST_SIZE_IN_WORDS** 12
- #define **CC_HASH_SHA512_DIGEST_SIZE_IN_WORDS** 16
- #define **CC_HASH_SHA224_DIGEST_SIZE_IN_BYTES** 28
- #define **CC_HASH_SHA256_DIGEST_SIZE_IN_BYTES** 32
- #define **CC_HASH_SHA384_DIGEST_SIZE_IN_BYTES** 48
- #define **CC_HASH_SHA512_DIGEST_SIZE_IN_BYTES** 64
- #define **CC_HASH_BLOCK_SIZE_IN_WORDS** 16
- #define **CC_HASH_BLOCK_SIZE_IN_BYTES** 64
- #define **CC_HASH_SHA512_BLOCK_SIZE_IN_WORDS** 32
- #define **CC_HASH_SHA512_BLOCK_SIZE_IN_BYTES** 128
- #define **CC_HASH_UPDATE_DATA_MAX_SIZE_IN_BYTES** (1 << 29)

2.6.38.15.3 Typedefs

- typedef uint32_t **CCHashResultBuf_t**[**CC_HASH_RESULT_SIZE_IN_WORDS**]
- typedef struct **CCHashUserContext_t CCHashUserContext_t**

2.6.38.15.4 Enumerations

- enum **CCHashOperationMode_t** { **CC_HASH_SHA1_mode** = 0, **CC_HASH_SHA224_mode** = 1, **CC_HASH_SHA256_mode** = 2, **CC_HASH_SHA384_mode** = 3, **CC_HASH_SHA512_mode** = 4, **CC_HASH_MD5_mode** = 5, **CC_HASH_NumOfModes**, **CC_HASH_OperationModeLast** = 0x7FFFFFFF }

2.6.38.16 cc_hash_defs_proj.h File Reference

This file contains HASH definitions.

2.6.38.16.1 Macros

- #define **CC_HASH_USER_CTX_SIZE_IN_WORDS** 197

2.6.38.17 cc_lib.h File Reference

This file contains all the enums and definitions that are used for the CryptoCell library initiation and finish APIs, as well as the APIs themselves.

```
#include "cc_pal_types.h"

#include "cc_chinese_cert.h"

#include "cc_cert_ctx.h"

#include "cc_axi_ctrl.h"
```

2.6.38.17.1 Enumerations

- enum **CClibRetCode_t** { **CC_LIB_RET_OK** = 0, **SA_SILIB_RET_ENODEV**, **SA_SILIB_RET_EINTERNAL**, **SA_SILIB_RET_ENOTSUP**, **SA_SILIB_RET_ENOPERM**, **SA_SILIB_RET_EINVAL**, **SA_SILIB_RET_HW_Q_INIT**, **SA_SILIB_RET_COMPLETION**, **CC_LIB_RET_HAL**, **CC_LIB_RET_EINVAL_PIDR**, **CC_LIB_RET_EINVAL_CIDR**, **SA_SILIB_RET_ASYM_ERR**, **CC_LIB_RET_RND_INST_ERR**, **CC_LIB_RET_EINVAL_HW_VERSION**, **CC_LIB_RET_EINVAL_HW_SIGNATURE**, **CC_LIB_RET_PAL**, **CC_LIB_INCORRECT_HW_VERSION_SLIM_VS_FULL**, **CC_LIB_RET_CACHE_PARAMS_ERROR**, **SA_SILIB_RET_ECHCERT**, **CC_LIB_RESERVE32B** = 0x7FFFFFFFL }

2.6.38.17.2 Functions

- **CClibRetCode_t CC_LibInit** (bool isChCertSupport, **CCCertKatContext_t** *pCertCtx, **CCAxiFields_t** *pAxiFields)

  This function performs global initialization of the Arm CryptoCell TEE runtime library; it must be called once per cold boot cycle. As part of the global initialization the function verifies that all the cryptographic engines are working as expected by running known answer tests. If a test fails (the function returns an error), it signifies that there is a fatal error, and it should be handled accordingly.

- void **CC_LibFini** (void)

This function finalizes the library operations. It calls HAL and PAL terminate functions.

2.6.38.17.3 Enumeration Type Documentation

enum **CClibRetCode_t**

Enumerator:

Enum	Description
CC_LIB_RET_OK	A success indication.
SA_SILIB_RET_ENODEV	Device not opened or does not exist.
SA_SILIB_RET_EINTERNAL	Internal driver error (check system log).
SA_SILIB_RET_ENOTSUP	Unsupported function or option.
SA_SILIB_RET_ENOPERM	Not enough permissions for request.
SA_SILIB_RET_EINVAL	Invalid parameters.
SA_SILIB_RET_HW_Q_INIT	Reserved.
SA_SILIB_RET_COMPLETION	Error in adaptor modules initialization.
CC_LIB_RET_HAL	Error in Hardware Adaption Layer initialization.
CC_LIB_RET_EINVAL_PIDR	Invalid peripheral ID.
CC_LIB_RET_EINVAL_CIDR	Invalid component ID.
SA_SILIB_RET_ASYM_ERR	Reserved.
CC_LIB_RET_RND_INST_ERR	Reserved.
CC_LIB_RET_EINVAL_HW_VERSION	Invalid HW version.
CC_LIB_RET_EINVAL_HW_SIGNATURE	Invalid HW signature.
CC_LIB_RET_PAL	Error in Platform Adaption Layer initialization.
CC_LIB_INCORRECT_HW_VERSION_SLIM_VS_FULL	Mismatched HW and SW products - SW is CC703, but HW is not.
CC_LIB_RET_CACHE_PARAMS_ERROR	Error setting the cache parameters due to invalid input parameter.
SA_SILIB_RET_ECHCERT	Chinese certification tests error.
CC_LIB_RESERVE32B	Reserved.

2.6.38.17.4 Function Documentation

void CC_LibFini (void)

Returns:

CC_LIB_RET_OK on success.

A non-zero value in case of failure.

CClibRetCode_t CC_LibInit (bool *isChCertSupport*, **CCCertKatContext_t** * *pCertCtx*, **CCAxiFields_t** * *pAxiFields*)

Unlike the other APIs in the library, this API is not thread-safe.

Returns:

CC_LIB_RET_OK on success.

A non-zero value in case of failure.

2.6.38.18 cc_pal_abort.h File Reference

This file includes all PAL APIs.

```
#include "cc_pal_abort_plat.h"
```

2.6.38.18.1 Functions

- void **CC_PalAbort** (const char *exp)

  This function performs the "Abort" operation. It must be implemented according to the specific platform and OS.

2.6.38.19 cc_pal_barrier.h File Reference

This file contains the definitions and APIs for memory-barrier implementation.

2.6.38.19.1 Functions

- void **CC_PalWmb** (void)
- void **CC_PalRmb** (void)

2.6.38.19.2 Detailed Description

This is a placeholder for platform-specific memory barrier implementation. The secure core driver should include a memory barrier, before and after the last word of the descriptor, to allow correct order between the words and different descriptors.

2.6.38.20 cc_pal_cert.h File Reference

This file contains definitions that are used by the CERT related APIs. The implementation of these functions need to be replaced according to the Platform and TEE_OS.

```
#include "cc_pal_types_plat.h"
```

2.6.38.20.1 Functions

- **CCError_t CC_PalCertGetState** (uint32_t *pCertState)

  This function purpose is to get the CERT state.

- **CCError_t CC_PalCertGetError** (uint32_t *pCertError)

  This function purpose is to get the CERT error.

- **CCError_t CC_PalCertGetTrace** (uint32_t *pCertTrace)

  This function purpose is to get the CERT trace.

- **CCError_t CC_PalCertSetState** (uint32_t certState)

  This function purpose is to set the CERT state.

- **CCError_t CC_PalCertSetError** (uint32_t certError)

  This function purpose is to set the CERT error.

- **CCError_t CC_PalCertSetTrace** (uint32_t certTrace)

  This function purpose is to set the CERT trace.

- **CCError_t CC_PalCertWaitForReeStatus** (void)

  This function purpose is to wait for CERT interrupt. After GPR0 (==CERT) interrupt is detected, clear the interrupt in ICR, and call CC_FipsIrqHandle.

- **CCError_t CC_PalCertStopWaitingRee** (void)

  This function purpose is to stop waiting for REE CERT interrupt. since TEE lib is terminating.

2.6.38.21 cc_pal_compiler.h File Reference

This file contains CryptoCell PAL platform-dependent compiler-related definitions.

2.6.38.21.1 Macros

- #define **CC_PAL_COMPILER_SECTION**(sectionName) __attribute__((section(sectionName)))
- #define **CC_PAL_COMPILER_KEEP_SYMBOL** __attribute__((used))
- #define **CC_PAL_COMPILER_ALIGN**(alignement) __attribute__((aligned(alignement)))
- #define **CC_PAL_COMPILER_FUNC_NEVER_RETURNS** __attribute__((noreturn))
- #define **CC_PAL_COMPILER_FUNC_DONT_INLINE** __attribute__((noinline))
- #define **CC_PAL_COMPILER_TYPE_MAY_ALIAS** __attribute__((__may_alias__))
- #define **CC_PAL_COMPILER_SIZEOF_STRUCT_MEMBER**(type_name, member_name) sizeof(((type_name *)0)->member_name)
- #define **CC_ASSERT_CONCAT_**(a, b) a##b

- #define **CC_ASSERT_CONCAT**(a, b) **CC_ASSERT_CONCAT_**(a, b)
- #define **CC_PAL_COMPILER_ASSERT**(cond, message) enum { **CC_ASSERT_CONCAT**(assert_line_, __LINE__) = 1/(!!(cond)) }

2.6.38.22 cc_pal_dma.h File Reference

This file contains definitions that are used for DMA-related APIs. The implementation of these functions need to be replaced according to the platform and OS.

```
#include "cc_pal_types.h"

#include "cc_pal_dma_plat.h"

#include "cc_pal_dma_defs.h"
```

2.6.38.22.1 Data Structures

- struct **CCPalDmaBlockInfo_t**

  User buffer scatter information.

2.6.38.22.2 Macros

- #define **SET_WORD_LE**

2.6.38.22.3 Functions

- uint32_t **CC_PalDmaBufferMap** (uint8_t *pDataBuffer, uint32_t buffSize, **CCPalDmaBufferDirection_t** copyDirection, uint32_t *pNumOfBlocks, **CCPalDmaBlockInfo_t** *pDmaBlockList, **CC_PalDmaBufferHandle** *dmaBuffHandle)

  This function is called by the CryptoCell runtime library before the HW is used. It maps a given data buffer (virtual address) for CryptoCell HW DMA use (physical address), and returns the list of one or more DMA-able (physical) blocks. Once it is called, only CryptoCell HW access to the buffer is allowed, until it is unmapped.

- uint32_t **CC_PalDmaBufferUnmap** (uint8_t *pDataBuffer, uint32_t buffSize, **CCPalDmaBufferDirection_t** copyDirection, uint32_t numOfBlocks, **CCPalDmaBlockInfo_t** *pDmaBlockList, **CC_PalDmaBufferHandle** dmaBuffHandle)

  This function is called by the CryptoCell runtime library after the HW is used. It unmaps a given buffer and frees its associated resources, if needed. It may unlock the buffer and flush it for CPU use. Once it is called, CryptoCell HW does not require any further access to this buffer.

- uint32_t **CC_PalDmaContigBufferAllocate** (uint32_t buffSize, uint8_t **ppVirtBuffAddr)

  Allocates a DMA-contiguous buffer for CPU use, and returns its virtual address. Before passing the buffer to the CryptoCell HW, **CC_PalDmaBufferMap** should be called.

- uint32_t **CC_PalDmaContigBufferFree** (uint32_t buffSize, uint8_t *pVirtBuffAddr)

  Frees resources previously allocated by **CC_PalDmaContigBufferAllocate**.

- uint32_t **CC_PalIsDmaBufferContiguous** (uint8_t *pDataBuffer, uint32_t buffSize)

Checks whether the buffer is guaranteed to be a single contiguous DMA block.

2.6.38.23 cc_pal_dma_defs.h File Reference

This file contains the platform-dependent DMA definitions.

2.6.38.23.1 Typedefs

- typedef void ***CC_PalDmaBufferHandle**

2.6.38.23.2 Enumerations

- enum **CCPalDmaBufferDirection_t** { **CC_PAL_DMA_DIR_NONE** = 0, **CC_PAL_DMA_DIR_TO_DEVICE** = 1, **CC_PAL_DMA_DIR_FROM_DEVICE** = 2, **CC_PAL_DMA_DIR_BI_DIRECTION** = 3, **CC_PAL_DMA_DIR_MAX**, **CC_PAL_DMA_DIR_RESERVE32** = 0x7FFFFFFF }

2.6.38.24 cc_pal_error.h File Reference

This file contains the error definitions of the platform-dependent PAL APIs.

2.6.38.24.1 Macros

- #define **CC_PAL_BASE_ERROR** 0x0F000000
- #define **CC_PAL_MEM_BUF1_GREATER CC_PAL_BASE_ERROR** + 0x01UL
- #define **CC_PAL_MEM_BUF2_GREATER CC_PAL_BASE_ERROR** + 0x02UL
- #define **CC_PAL_SEM_CREATE_FAILED CC_PAL_BASE_ERROR** + 0x03UL
- #define **CC_PAL_SEM_DELETE_FAILED CC_PAL_BASE_ERROR** + 0x04UL
- #define **CC_PAL_SEM_WAIT_TIMEOUT CC_PAL_BASE_ERROR** + 0x05UL
- #define **CC_PAL_SEM_WAIT_FAILED CC_PAL_BASE_ERROR** + 0x06UL
- #define **CC_PAL_SEM_RELEASE_FAILED CC_PAL_BASE_ERROR** + 0x07UL
- #define **CC_PAL_ILLEGAL_ADDRESS CC_PAL_BASE_ERROR** + 0x08UL

2.6.38.25 cc_pal_init.h File Reference

This file contains the PAL layer entry point. It includes the definitions and APIs for PAL initialization and termination.

```
#include "cc_pal_types.h"
```

2.6.38.25.1 Functions

- int **CC_PalInit** (void)

This function performs all initializations that may be required by your PAL implementation, specifically by the DMA-able buffer scheme.

- void **CC_PalTerminate** (void)

  This function terminates the PAL implementation and frees the resources that were allocated by **CC_PalInit**.

2.6.38.26 cc_pal_log.h File Reference

This file contains the PAL layer log definitions. The log is disabled by default.

```
#include "cc_pal_types.h"

#include "cc_pal_log_plat.h"
```

2.6.38.26.1 Macros

- #define **CC_PAL_LOG_LEVEL_NULL** (-1)
- #define **CC_PAL_LOG_LEVEL_ERR** 0
- #define **CC_PAL_LOG_LEVEL_WARN** 1
- #define **CC_PAL_LOG_LEVEL_INFO** 2
- #define **CC_PAL_LOG_LEVEL_DEBUG** 3
- #define **CC_PAL_LOG_LEVEL_TRACE** 4
- #define **CC_PAL_LOG_LEVEL_DATA** 5
- #define **CC_PAL_LOG_CUR_COMPONENT** 0xFFFFFFFF
- #define **CC_PAL_LOG_CUR_COMPONENT_NAME** "CC"
- #define **CC_PAL_MAX_LOG_LEVEL CC_PAL_LOG_LEVEL_NULL**
- #define **__CC_PAL_LOG_LEVEL_EVAL**(level) level
- #define **_CC_PAL_MAX_LOG_LEVEL __CC_PAL_LOG_LEVEL_EVAL**(**CC_PAL_MAX_LOG_LEVEL**)
- #define **_CC_PAL_LOG**(level, format, ...)
- #define **CC_PAL_LOG_ERR**(...) do {} while (0)
- #define **CC_PAL_LOG_WARN**(...) do {} while (0)
- #define **CC_PAL_LOG_INFO**(...) do {} while (0)
- #define **CC_PAL_LOG_DEBUG**(...) do {} while (0)
- #define **CC_PAL_LOG_DUMP_BUF**(msg, buf, size) do {} while (0)
- #define **CC_PAL_LOG_TRACE**(...) do {} while (0)
- #define **CC_PAL_LOG_DATA**(...) do {} while (0)

2.6.38.27 cc_pal_mem.h File Reference

This file contains functions for memory operations.

```
#include "cc_pal_types.h"

#include "cc_pal_mem_plat.h"

#include "cc_pal_malloc_plat.h"

#include <stdlib.h>

#include <string.h>
```

2.6.38.27.1 Macros

- #define **CC_PalMemCmp**(aTarget, aSource, aSize) CC_PalMemCmpPlat(aTarget, aSource, aSize)

  This function compares between two given buffers, according to the given size.

- #define **CC_PalMemCopy**(aDestination, aSource, aSize) CC_PalMemCopyPlat(aDestination, aSource, aSize)

  This function copies aSize bytes from the source buffer to the destination buffer.

- #define **CC_PalMemMove**(aDestination, aSource, aSize) CC_PalMemMovePlat(aDestination, aSource, aSize)

  This function moves aSize bytes from the source buffer to the destination buffer. This function supports overlapped buffers.

- #define **CC_PalMemSet**(aTarget, aChar, aSize) CC_PalMemSetPlat(aTarget, aChar, aSize)

  This function sets aSize bytes of aChar in the given buffer.

- #define **CC_PalMemSetZero**(aTarget, aSize) CC_PalMemSetZeroPlat(aTarget, aSize)

  This function sets aSize bytes in the given buffer with zeroes.

- #define **CC_PalMemMalloc**(aSize) CC_PalMemMallocPlat(aSize)

  This function allocates a memory buffer according to aSize .

- #define **CC_PalMemRealloc**(aBuffer, aNewSize) CC_PalMemReallocPlat(aBuffer, aNewSize)

  This function reallocates a memory buffer according to aNewSize . The contents of the old buffer is moved to the new location.

- #define **CC_PalMemFree**(aBuffer) CC_PalMemFreePlat(aBuffer)

  This function frees a previously-allocated buffer.

2.6.38.27.2 Detailed Description

The functions are generally implemented as wrappers to different operating-system calls.

None of the described functions validate the input parameters, so that the behavior of the APIs in case of an illegal parameter is dependent on the behavior of the operating system.

2.6.38.28 cc_pal_memmap.h File Reference

This file contains functions for memory mapping.

```
#include "cc_pal_types.h"

#include "cc_address_defs.h"
```

2.6.38.28.1 Functions

- uint32_t **CC_PalMemMap** (CCDmaAddr_t physicalAddress, uint32_t mapSize, uint32_t **ppVirtBuffAddr)

  This function returns the base virtual address that maps the base physical address.

- uint32_t **CC_PalMemUnMap** (uint32_t *pVirtBuffAddr, uint32_t mapSize)

  This function unmaps a specified address range that was previously mapped by **CC_PalMemMap**.

2.6.38.28.2 Detailed Description

None of the described functions validate the input parameters, so that the behavior of the APIs in case of an illegal parameter is dependent on the behavior of the operating system.

2.6.38.29 cc_pal_mutex.h File Reference

This file contains functions for resource management (mutex operations).

```
#include "cc_pal_mutex_plat.h"

#include "cc_pal_types_plat.h"
```

2.6.38.29.1 Functions

- **CCError_t CC_PalMutexCreate** (CC_PalMutex *pMutexId)

  This function creates a mutex.

- **CCError_t CC_PalMutexDestroy** (CC_PalMutex *pMutexId)

  This function destroys a mutex.

- **CCError_t CC_PalMutexLock** (CC_PalMutex *pMutexId, uint32_t aTimeOut)

This function waits for a mutex with aTimeOut . aTimeOut is specified in milliseconds. A value of aTimeOut=CC_INFINITE means that the function will not return.

- **CCError_t CC_PalMutexUnlock** (CC_PalMutex *pMutexId)

  This function releases the mutex.

2.6.38.29.2 Detailed Description

These functions are generally implemented as wrappers to different operating-system calls.

None of the described functions validate the input parameters, so that the behavior of the APIs in case of an illegal parameter is dependent on the behavior of the operating system.

2.6.38.30 cc_pal_pm.h File Reference

This file contains the definitions and APIs for power-management implementation.

2.6.38.30.1 Functions

- void **CC_PalPowerDown** (void)

  This function powers down CryptoCell.

- void **CC_PalPowerUp** (void)

  This function powers up CryptoCell.

2.6.38.30.2 Detailed Description

This is a placeholder for platform-specific power management implementation. The module should be updated whether CryptoCell is active or not, to notify the external PMU when it might be powered down.

2.6.38.31 cc_pal_trng.h File Reference

This file contains APIs for retrieving TRNG user parameters.

```
#include "cc_pal_types.h"
```

2.6.38.31.1 Data Structures

- struct **CC_PalTrngParams_t**

2.6.38.31.2 Functions

- CCError_t CC_PalTrngParamGet (CC_PalTrngParams_t *pTrngParams, size_t *pParamsSize)

  This function returns the TRNG user parameters.

2.6.38.32 cc_pal_types.h File Reference

This file contains platform-dependent definitions and types of the PAL layer.

```
#include "cc_pal_types_plat.h"
```

2.6.38.32.1 Macros

- #define CC_SUCCESS 0UL
- #define CC_FAIL 1UL
- #define CC_OK 0
- #define CC_UNUSED_PARAM(prm) ((void)prm)
- #define CC_MAX_UINT32_VAL (0xFFFFFFFF)
- #define CC_MIN(a, b) (((a) < (b)) ? (a): (b))
- #define CC_MAX(a, b) (((a) > (b)) ? (a): (b))
- #define CALC_FULL_BYTES(numBits) ((numBits)/CC_BITS_IN_BYTE + (((numBits) & (CC_BITS_IN_BYTE-1)) > 0))
- #define CALC_FULL_32BIT_WORDS(numBits) ((numBits)/CC_BITS_IN_32BIT_WORD + (((numBits) & (CC_BITS_IN_32BIT_WORD-1)) > 0))
- #define CALC_32BIT_WORDS_FROM_BYTES(sizeBytes) ((sizeBytes)/CC_32BIT_WORD_SIZE + (((sizeBytes) & (CC_32BIT_WORD_SIZE-1)) > 0))
- #define CALC_32BIT_WORDS_FROM_64BIT_DWORD(sizeWords) (sizeWords *CC_32BIT_WORD_IN_64BIT_DWORD)
- #define ROUNDUP_BITS_TO_32BIT_WORD(numBits) (CALC_FULL_32BIT_WORDS(numBits) *CC_BITS_IN_32BIT_WORD)
- #define ROUNDUP_BITS_TO_BYTES(numBits) (CALC_FULL_BYTES(numBits) *CC_BITS_IN_BYTE)
- #define ROUNDUP_BYTES_TO_32BIT_WORD(sizeBytes) (CALC_32BIT_WORDS_FROM_BYTES(sizeBytes) *CC_32BIT_WORD_SIZE)
- #define CALC_WORDS_TO_BYTES(numwords) ((numwords)*CC_32BIT_WORD_SIZE)
- #define CC_1K_SIZE_IN_BYTES 1024
- #define CC_BITS_IN_BYTE 8
- #define CC_BITS_IN_32BIT_WORD 32
- #define CC_32BIT_WORD_SIZE 4
- #define CC_32BIT_WORD_IN_64BIT_DWORD 2

2.6.38.32.2 Enumerations

- enum **CCBool** { **CC_FALSE** = 0, **CC_TRUE** = 1 }

2.6.38.33 cc_pal_types_plat.h File Reference

This file contains basic platform-dependent type definitions.

```
#include <stdint.h>

#include <stddef.h>

#include <stdbool.h>
```

2.6.38.33.1 Macros

- #define **CCError_t CCStatus**
- #define **CC_INFINITE** 0xFFFFFFFFUL
- #define **CEXPORT_C**
- #define **CIMPORT_C**

2.6.38.33.2 Typedefs

- typedef uintptr_t **CCVirtAddr_t**
- typedef uint32_t **CCBool_t**
- typedef uint32_t **CCStatus**

2.6.38.33.3 Macro Definition Documentation

#define CC_INFINITE 0xFFFFFFFFUL

Defines an unlimited (infinite) time frame.

#define CCError_t **CCStatus**

Defines error return.

#define CEXPORT_C

Defines for C export.

#define CIMPORT_C

Defines for C import.

2.6.38.33.4 Typedef Documentation

typedef uint32_t **CCBool_t**

Defines for boolean variable.

typedef uint32_t **CCStatus**

Defines for return status.

typedef uintptr_t **CCVirtAddr_t**

Defines for virtual address.

2.6.38.34 cc_pka_defs_hw.h File Reference

This file contains all the enums and definitions that are used in the PKA related code.

```
#include "cc_pal_types.h"
```

```
#include "cc_pka_hw_plat_defs.h"
```

2.6.38.34.1 Macros

- #define CC_RSA_MAXIMUM_MOD_BUFFER_SIZE_IN_WORDS ((CC_RSA_MAX_VALID_KEY_SIZE_VALUE_IN_BITS + CC_PKA_WORD_SIZE_IN_BITS) / CC_BITS_IN_32BIT_WORD)
- #define CC_ECPKI_MODUL_MAX_LENGTH_IN_BITS 521
- #define CC_PKA_BARRETT_MOD_TAG_BUFF_SIZE_IN_WORDS 5
- #define CC_PKA_ECPKI_BARRETT_MOD_TAG_BUFF_SIZE_IN_WORDS CC_PKA_BARRETT_MOD_TAG_BUFF_SIZE_IN_WORDS
- #define CC_PKA_BARRETT_MOD_TAG_SIZE_IN_WORDS (((CC_PKA_WORD_SIZE_IN_BITS + PKA_EXTRA_BITS - 1) + (CC_BITS_IN_32BIT_WORD - 1)) / CC_BITS_IN_32BIT_WORD)
- #define CC_PKA_MAXIMUM_MOD_BUFFER_SIZE_IN_WORDS CC_RSA_MAXIMUM_MOD_BUFFER_SIZE_IN_WORDS
- #define CC_PKA_PUB_KEY_BUFF_SIZE_IN_WORDS (2*CC_PKA_BARRETT_MOD_TAG_BUFF_SIZE_IN_WORDS)
- #define CC_PKA_PRIV_KEY_BUFF_SIZE_IN_WORDS (2*CC_PKA_BARRETT_MOD_TAG_BUFF_SIZE_IN_WORDS)
- #define CC_PKA_KGDATA_BUFF_SIZE_IN_WORDS (3*CC_PKA_BARRETT_MOD_TAG_BUFF_SIZE_IN_WORDS + 3*CC_PKA_MAXIMUM_MOD_BUFFER_SIZE_IN_WORDS)
- #define CC_ECPKI_MODUL_MAX_LENGTH_IN_WORDS 18
- #define CC_ECPKI_ORDER_MAX_LENGTH_IN_WORDS (CC_ECPKI_MODUL_MAX_LENGTH_IN_WORDS + 1)
- #define CC_PKA_DOMAIN_BUFF_SIZE_IN_WORDS (2*CC_PKA_BARRETT_MOD_TAG_BUFF_SIZE_IN_WORDS)
- #define COUNT_NAF_WORDS_PER_KEY_WORD 8
- #define CC_PKA_ECDSA_NAF_BUFF_MAX_LENGTH_IN_WORDS (COUNT_NAF_WORDS_PER_KEY_WORD*CC_ECPKI_ORDER_MAX_LENGTH_IN_WORDS + 1)
- #define CC_PKA_ECPKI_SCALAR_MUL_BUFF_MAX_LENGTH_IN_WORDS (CC_PKA_ECDSA_NAF_BUFF_MAX_LENGTH_IN_WORDS + CC_ECPKI_MODUL_MAX_LENGTH_IN_WORDS + 2)

- #define CC_PKA_ECPKI_BUILD_TMP_BUFF_MAX_LENGTH_IN_WORDS (3*CC_ECPKI_MODUL_MAX_LENGTH_IN_WORDS+CC_PKA_ECPKI_SCALAR_MUL_BUFF_MAX_LENGTH_IN_WORDS)
- #define CC_PKA_ECDSA_SIGN_BUFF_MAX_LENGTH_IN_WORDS (6*CC_ECPKI_MODUL_MAX_LENGTH_IN_WORDS+CC_PKA_ECPKI_SCALAR_MUL_BUFF_MAX_LENGTH_IN_WORDS)
- #define CC_PKA_ECDH_BUFF_MAX_LENGTH_IN_WORDS (2*CC_ECPKI_ORDER_MAX_LENGTH_IN_WORDS + CC_PKA_ECPKI_SCALAR_MUL_BUFF_MAX_LENGTH_IN_WORDS)
- #define CC_PKA_KG_BUFF_MAX_LENGTH_IN_WORDS (2*CC_ECPKI_ORDER_MAX_LENGTH_IN_WORDS + CC_PKA_ECPKI_SCALAR_MUL_BUFF_MAX_LENGTH_IN_WORDS)
- #define CC_PKA_ECDSA_VERIFY_BUFF_MAX_LENGTH_IN_WORDS (3*CC_ECPKI_MODUL_MAX_LENGTH_IN_WORDS)

2.6.38.35 cc_pka_hw_plat_defs.h File Reference

Contains the enums and definitions that are used in the PKA code.

```
#include "cc_pal_types.h"
```

2.6.38.35.1 Macros

- #define CC_PKA_WORD_SIZE_IN_BITS 128
- #define CC_RSA_MAX_VALID_KEY_SIZE_VALUE_IN_BITS 4096
- #define CC_RSA_MAX_KEY_GENERATION_HW_SIZE_BITS 4096
- #define BSV_CERT_RSA_KEY_SIZE_IN_BITS 2048
- #define BSV_CERT_RSA_KEY_SIZE_IN_BYTES (BSV_CERT_RSA_KEY_SIZE_IN_BITS/CC_BITS_IN_BYTE)
- #define BSV_CERT_RSA_KEY_SIZE_IN_WORDS (BSV_CERT_RSA_KEY_SIZE_IN_BITS/CC_BITS_IN_32BIT_WORD)
- #define PKA_EXTRA_BITS 8
- #define PKA_MAX_COUNT_OF_PHYS_MEM_REGS 32
- #define RSA_PKA_BARRETT_MOD_TAG_BUFF_SIZE_IN_WORDS 5
- #define RSA_PKA_BARRETT_MOD_TAG_BUFF_SIZE_IN_BYTES (RSA_PKA_BARRETT_MOD_TAG_BUFF_SIZE_IN_WORDS*CC_32BIT_WORD_SIZE)

2.6.38.36 cc_regs.h File Reference

This file contains macro definitions for accessing Arm CryptoCell's registers.

```
#include "cc_bitops.h"
```

```
#include "dx_reg_base_host.h"

#include "cc_registers.h"
```

2.6.38.36.1 Macros

- #define **SB_REG_ADDR**(base, reg_name) (base + CC_REG_OFFSET(CRY_KERNEL, reg_name))
- #define **SB_REG_ADDR_UNIT**(base, reg_name, unit) (base + CC_REG_OFFSET(unit, reg_name))
- #define **CC_REG_OFFSET**(unit_name, reg_name) (CC_BASE_ ## unit_name + CC_ ## reg_name ## _REG_OFFSET)
- #define **CC_REG_BIT_SHIFT**(reg_name, field_name) (CC_ ## reg_name ## _ ## field_name ## _BIT_SHIFT)
- #define **CC_REG_BIT_MASK**(reg_name, field_name) (**BITMASK**(CC_ ## reg_name ## _ ## field_name ## _BIT_SIZE) << (CC_ ## reg_name ## _ ## field_name ## _BIT_SHIFT))
- #define **CC_REG_BIT_SIZE**(reg_name, field_name) (CC_ ## reg_name ## _ ## field_name ## _BIT_SIZE)
- #define **CC_REG_FLD_GET**(unit_name, reg_name, fld_name, reg_val)
- #define **CC_REG_FLD_GET2**(unit_name, reg_name, fld_name, reg_val)
- #define **CC_REG_FLD_SET**(unit_name, reg_name, fld_name, reg_shadow_var, new_fld_val)

2.6.38.37 cc_rnd_common.h File Reference

This file contains the CryptoCell random-number generation APIs.

```
#include "cc_error.h"

#include "cc_aes_defs.h"

#include "cc_rnd_common_trng.h"
```

2.6.38.37.1 Data Structures

- struct **CCRndState_t**

  The structure for the RND state. This includes internal data that must be saved by the user between boots.

- struct **CCRndContext_t**

2.6.38.37.2 Macros

- #define **CC_RND_SEED_MAX_SIZE_WORDS** 12
- #define **CC_RND_ADDITINAL_INPUT_MAX_SIZE_WORDS CC_RND_SEED_MAX_SIZE_WORDS**
- #define **CC_RND_MAX_GEN_VECTOR_SIZE_BITS** 0x7FFFF

- #define **CC_RND_MAX_GEN_VECTOR_SIZE_BYTES** 0xFFFF
- #define **CC_RND_REQUESTED_SIZE_COUNTER** 0x3FFFF

2.6.38.37.3 Typedefs

- typedef int(***CCRndGenerateVectWorkFunc_t**) (void *rndState_ptr, unsigned char *out_ptr, size_t outSizeBytes)

2.6.38.37.4 Functions

- **CIMPORT_C CCError_t CC_RndGenerateVectorInRange** (**CCRndGenerateVectWorkFunc_t** f_rng, void *p_rng, size_t rndSizeInBits, uint8_t *maxVect_ptr, uint8_t *rndVect_ptr)

  Generates a random vector with specific limitations by testing candidates (described and used in FIPS Publication 186-4: Digital Signature Standard (DSS): B.1.2, B.4.2 etc.).

2.6.38.37.5 Detailed Description

The random-number generation module implements *NIST Special Publication 800-90A: Recommendation for Random Number Generation Using Deterministic Random Bit Generators.*

2.6.38.38 cc_rnd_common_trng.h File Reference

This file contains the CryptoCell true-random-number generation definitions. The true-random-number generation module defines the database used for the TRNG operations.

```
#include "cc_error.h"

#include "cc_pal_types_plat.h"

#include "cc_pal_trng.h"
```

2.6.38.38.1 Data Structures

- struct **CCTrngWorkBuff_t**
- struct **CCTrngParams_t**
- struct **CCTrngState_t**

2.6.38.38.2 Macros

- #define **CC_TRNG_WORK_BUFFER_SIZE_WORDS** 136
- #define **CC_RND_TRNG_SRC_INNER_OFFSET_WORDS** 2
- #define **CC_RND_TRNG_SRC_INNER_OFFSET_BYTES** (**CC_RND_TRNG_SRC_INNER_OFFSET_WORDS***sizeof(uint32_t))

2.6.38.38.3 Typedefs

- typedef struct **CCTrngWorkBuff_t CCTrngWorkBuff_t**

- typedef struct **CCTrngParams_t CCTrngParams_t**
- typedef struct **CCTrngState_t CCTrngState_t**

2.6.38.39 cc_rnd_error.h File Reference

This file contains the definitions of the CryptoCell RND errors.

```
#include "cc_error.h"
```

2.6.38.39.1 Macros

- #define **CC_RND_DATA_OUT_POINTER_INVALID_ERROR** (**CC_RND_MODULE_ERROR_BASE** + 0x0UL)
- #define **CC_RND_CAN_NOT_GENERATE_RAND_IN_RANGE** (**CC_RND_MODULE_ERROR_BASE** + 0x1UL)
- #define **CC_RND_CPRNG_TEST_FAIL_ERROR** (**CC_RND_MODULE_ERROR_BASE** + 0x2UL)
- #define **CC_RND_ADDITIONAL_INPUT_BUFFER_NULL** (**CC_RND_MODULE_ERROR_BASE** + 0x3UL)
- #define **CC_RND_ADDITIONAL_INPUT_SIZE_ERROR** (**CC_RND_MODULE_ERROR_BASE** + 0x4UL)
- #define **CC_RND_DATA_SIZE_OVERFLOW_ERROR** (**CC_RND_MODULE_ERROR_BASE** + 0x5UL)
- #define **CC_RND_VECTOR_SIZE_ERROR** (**CC_RND_MODULE_ERROR_BASE** + 0x6UL)
- #define **CC_RND_RESEED_COUNTER_OVERFLOW_ERROR** (**CC_RND_MODULE_ERROR_BASE** + 0x7UL)
- #define **CC_RND_INSTANTIATION_NOT_DONE_ERROR** (**CC_RND_MODULE_ERROR_BASE** + 0x8UL)
- #define **CC_RND_TRNG_LOSS_SAMPLES_ERROR** (**CC_RND_MODULE_ERROR_BASE** + 0x9UL)
- #define **CC_RND_TRNG_TIME_EXCEED_ERROR** (**CC_RND_MODULE_ERROR_BASE** + 0xAUL)
- #define **CC_RND_TRNG_LOSS_SAMPLES_AND_TIME_EXCEED_ERROR** (**CC_RND_MODULE_ERROR_BASE** + 0xBUL)
- #define **CC_RND_IS_KAT_MODE_ERROR** (**CC_RND_MODULE_ERROR_BASE** + 0xCUL)
- #define **CC_RND_OPERATION_IS_NOT_SUPPORTED_ERROR** (**CC_RND_MODULE_ERROR_BASE** + 0xDUL)
- #define **CC_RND_STATE_VALIDATION_TAG_ERROR** (**CC_RND_MODULE_ERROR_BASE** + 0xEUL)
- #define **CC_RND_IS_NOT_SUPPORTED** (**CC_RND_MODULE_ERROR_BASE** + 0xFUL)
- #define **CC_RND_GEN_VECTOR_FUNC_ERROR** (**CC_RND_MODULE_ERROR_BASE** + 0x14UL)

- #define CC_RND_WORK_BUFFER_PTR_INVALID_ERROR (CC_RND_MODULE_ERROR_BASE + 0x20UL)
- #define CC_RND_ILLEGAL_AES_KEY_SIZE_ERROR (CC_RND_MODULE_ERROR_BASE + 0x21UL)
- #define CC_RND_ILLEGAL_DATA_PTR_ERROR (CC_RND_MODULE_ERROR_BASE + 0x22UL)
- #define CC_RND_ILLEGAL_DATA_SIZE_ERROR (CC_RND_MODULE_ERROR_BASE + 0x23UL)
- #define CC_RND_ILLEGAL_PARAMETER_ERROR (CC_RND_MODULE_ERROR_BASE + 0x24UL)
- #define CC_RND_STATE_PTR_INVALID_ERROR (CC_RND_MODULE_ERROR_BASE + 0x25UL)
- #define CC_RND_TRNG_ERRORS_ERROR (CC_RND_MODULE_ERROR_BASE + 0x26UL)
- #define CC_RND_CONTEXT_PTR_INVALID_ERROR (CC_RND_MODULE_ERROR_BASE + 0x27UL)
- #define CC_RND_VECTOR_OUT_PTR_ERROR (CC_RND_MODULE_ERROR_BASE + 0x30UL)
- #define CC_RND_VECTOR_OUT_SIZE_ERROR (CC_RND_MODULE_ERROR_BASE + 0x31UL)
- #define CC_RND_MAX_VECTOR_IS_TOO_SMALL_ERROR (CC_RND_MODULE_ERROR_BASE + 0x32UL)
- #define CC_RND_KAT_DATA_PARAMS_ERROR (CC_RND_MODULE_ERROR_BASE + 0x33UL)
- #define CC_RND_TRNG_KAT_NOT_SUPPORTED_ERROR (CC_RND_MODULE_ERROR_BASE + 0x34UL)
- #define CC_RND_SRAM_NOT_SUPPORTED_ERROR (CC_RND_MODULE_ERROR_BASE + 0x35UL)
- #define CC_RND_AES_ERROR (CC_RND_MODULE_ERROR_BASE + 0x36UL)
- #define CC_RND_MODE_MISMATCH_ERROR (CC_RND_MODULE_ERROR_BASE + 0x37UL)

2.6.38.40 cc_sm2.h File Reference

This file defines the APIs that support the SM2 functions.

```
#include "cc_error.h"

#include "cc_ecpki_types.h"

#include "cc_rnd_common.h"

#include "cc_sm3_defs.h"

#include "cc_pal_types.h"
```

2.6.38.40.1 Data Structures

- struct **CCSm2FipsKatContext_t**
- struct **CCSm2KeyGenCHCertContext_t**
- struct **CC_Sm2KeContext_t**
- #define **CC_SM2_MODULE_LENGTH_IN_WORDS** 8
- #define **CC_SM2_ORDER_LENGTH_IN_WORDS** 8
- #define **CC_SM2_MODULE_LENGTH_IN_BYTES** 32
- #define **CC_SM2_ORDER_LENGTH_IN_BYTES** 32
- #define **CC_SM2_MAX_ID_LEN_IN_BITS** 65535
- #define **CC_SM2_MAX_ID_LEN_IN_BYTES CC_SM2_MAX_ID_LEN_IN_BITS** / **CC_BITS_IN_BYTE**
- #define **CC_SM2_MAX_MESSEGE_LEN** (1UL << 29)
- #define **CC_SM2_SIGNATURE_LENGTH_IN_BYTES CC_SM2_ORDER_LENGTH_IN_BYTES** *2
- #define **CC_SM2_CONF_VALUE_LENGTH_IN_BYTES** CC_SM3_RESULT_SIZE_IN_BYTES
- #define **CERT_SM2_DEFAULT_INPUT_AND_ID_SIZE** 32
- typedef struct **CCSm2FipsKatContext_t CCSm2FipsKatContext_t**
- typedef struct **CCSm2KeyGenCHCertContext_t CCSm2KeyGenCHCertContext_t**
- typedef struct **CC_Sm2KeContext_t CC_Sm2KeContext_t**
- **CIMPORT_C CCError_t CC_Sm2Sign** (**CCRndGenerateVectWorkFunc_t** f_rng, void *p_rng, const **CCEcpkiUserPrivKey_t** *pSm2PrivKey, const uint32_t *pHashInput, const size_t HashInputSize, uint8_t *pSignatureOut, size_t *pSignatureOutSize)

  This function performs an SM2 sign operation.
- **CIMPORT_C CCError_t CC_Sm2Verify** (const **CCEcpkiUserPublKey_t** *pUserPublKey, uint8_t *pSignatureIn, const size_t SignatureSizeBytes, const uint32_t *pHashInput, const size_t HashInputSize)

  This function performs an SM2 verify operation in integrated form.
- **CIMPORT_C CCError_t CC_Sm2ComputeMessageDigest** (const **CCEcpkiUserPublKey_t** *pUserPublKey, const char *pId, const size_t idlen, const uint8_t *pMsg, const size_t msglen, uint8_t *pWorkingBuffer, const size_t wblen, uint32_t *pOut, size_t *pOutlen)

  This function calculates both the ID digest and the message digest.
- **CIMPORT_C CCError_t CC_Sm2KeyExchangeContext_init** (**CC_Sm2KeContext_t** *pCtx, uint8_t *pWorkingBuffer, const size_t wblen, **CCEcpkiUserPublKey_t** *pPubKey, **CCEcpkiUserPrivKey_t** *pPrivKey, **CCEcpkiUserPublKey_t** *pRemoteUserPubKey, const char *pId, size_t idlen, const char *pRemoteId, size_t remoteIdLen, uint8_t is_initiator, uint8_t conf_required)

  The context initiation.
- **CIMPORT_C** void **CC_Sm2KeyExchangeContext_cleanup** (**CC_Sm2KeContext_t** *pCtx)

The context cleanup.

- **CIMPORT_C CCError_t CC_Sm2Kdf** (const **CC_Sm2KeContext_t** *pCtx, const size_t SharedSecretSizeInBits, uint8_t *pKeyOut, size_t *pKeyOutSize)

  The KDF.

- **CIMPORT_C CCError_t CC_Sm2CalculateECPoint** (**CCRndGenerateVectWorkFunc_t** f_rng, void *p_rng, **CC_Sm2KeContext_t** *pCtx, **CCEcpkiUserPublKey_t** *pRandomPoint)

  Calculates a random ECPoint.

- **CIMPORT_C CCError_t CC_Sm2CalculateSharedSecret** (**CC_Sm2KeContext_t** *pCtx, const **CCEcpkiUserPublKey_t** *pRandomPoint, uint8_t *pConfirmationValueOut, size_t *pConfirmationValueOutSize)

  Calculates shared secret and optionally the internal confirmation value and stores them into the context. Optionally calculates output confirmation value.

- **CIMPORT_C CCError_t CC_Sm2Confirmation** (const **CC_Sm2KeContext_t** *pCtx, const uint8_t *pConfirmationValue, const size_t confirmationValueSize)

  Verifies the confirmation value sent by other side with the one calculated and stored in the context

2.6.38.40.2 Macro Definition Documentation

#define CC_SM2_CONF_VALUE_LENGTH_IN_BYTES CC_SM3_RESULT_SIZE_IN_BYTES

SM2- Confirmation value size in bytes.

#define CC_SM2_MAX_ID_LEN_IN_BITS 65535

SM2 - Max length of ID in bytes.

#define CC_SM2_MAX_ID_LEN_IN_BYTES **CC_SM2_MAX_ID_LEN_IN_BITS** / **CC_BITS_IN_BYTE**

SM2 - Max length of ID in bytes.

#define CC_SM2_MAX_MESSEGE_LEN (1UL << 29)

SM2 - Max length of message in bytes.

#define CC_SM2_MODULE_LENGTH_IN_BYTES 32

SM2 - Length of the module in bytes.

#define CC_SM2_MODULE_LENGTH_IN_WORDS 8

SM2 - Length of the module in words.

#define CC_SM2_ORDER_LENGTH_IN_BYTES 32

SM2 - Length of the base point order in bytes.

#define CC_SM2_ORDER_LENGTH_IN_WORDS 8

SM2- Length of the the base point order in words.

#define CC_SM2_SIGNATURE_LENGTH_IN_BYTES **CC_SM2_ORDER_LENGTH_IN_BYTES** *2

SM2 - Signature output size in bytes.

#define CERT_SM2_DEFAULT_INPUT_AND_ID_SIZE 32

SM2- Max size of input and ID - chosen based on implementation of certification KAT tests.

2.6.38.40.3 Typedef Documentation

typedef struct **CC_Sm2KeContext_t CC_Sm2KeContext_t**

A structure to define key exchange context. All byte arrays in this structure are stored in the big endian byte ordering, and all word arrays are in the little endian byte and word ordering.

typedef struct **CCSm2FipsKatContext_t CCSm2FipsKatContext_t**

SM2 self-test data structure for Chinese certification.

typedef struct **CCSm2KeyGenCHCertContext_t CCSm2KeyGenCHCertContext_t**

SM2 self-test data structure for certification.

2.6.38.40.4 Function Documentation

CIMPORT_C CCError_t CC_Sm2CalculateECPoint (**CCRndGenerateVectWorkFunc_t** *f_rng*, void * *p_rng*, **CC_Sm2KeContext_t** * *pCtx*, **CCEcpkiUserPublKey_t** * *pRandomPoint*)

Returns:

`CC_OK` on success.

A non-zero value on failure

Parameters:

I/O	Parameter	Description
in	`f_rng`	A pointer to DRBG function
in,out	`p_rng`	A pointer to the random context - the input to f_rng.
in,out	`pCtx`	A pointer to a KE context
out	`pRandomPoint`	The output random EC point as an ephemeral public key.

CIMPORT_C CCError_t CC_Sm2CalculateSharedSecret (**CC_Sm2KeContext_t** * *pCtx*, const **CCEcpkiUserPublKey_t** * *pRandomPoint*, uint8_t * *pConfirmationValueOut*, size_t * *pConfirmationValueOutSize*)

Returns:

`CC_OK` on success.

A non-zero value on failure

Parameters:

I/O	Parameter	Description
in,out	`pCtx`	A pointer to the key exchange context.
in	`pRandomPoint`	A pointer to the random point from the second party.
out	`pConfirmationValueOut`	The output confirmation value.
in,out	`pConfirmationValueOutSize`	A pointer to the output confirmation value size in bytes

CIMPORT_C CCError_t CC_Sm2ComputeMessageDigest (const **CCEcpkiUserPublKey_t** * *pUserPublKey*, const char * *pId*, const size_t *idlen*, const uint8_t * *pMsg*, const size_t *msglen*, uint8_t * *pWorkingBuffer*, const size_t *wblen*, uint32_t * *pOut*, size_t * *pOutlen*)

Returns:

`CC_OK` on success.

A non-zero value on failure as defined **cc_ecpki_error.h** or **cc_sm3_error.h**.

Parameters:

I/O	Parameter	Description
in	`pUserPublKey`	A pointer to the public key
in	`pId`	A pointer to the ID.
in	`idlen`	The size of ID in bytes.
in	`pMsg`	A pointer to the message.
in	`msglen`	The size of the message in bytes.
in	`pWorkingBuffer`	The working buffer
in	`wblen`	The working buffer size should be at least 2 + modSizeInBytes*4 + ordSizeInBytes*2 idlen + msglen
out	`pOut`	A pointer to a buffer for the output.
in,out	`pOutlen`	A pointer to the output length in words.

CIMPORT_C CCError_t CC_Sm2Confirmation (const **CC_Sm2KeContext_t** * *pCtx*, const uint8_t * *pConfirmationValue*, const size_t *confirmationValueSize*)

Returns:

`CC_OK` on success.

A non-zero value on failure

Parameters:

I/O	Parameter	Description
in,out	`pCtx`	Pointer to the key exchange context.
in	`pConfirmationValue`	A pointer to a second party confirmation value.
in,out	`confirmationValueSize`	Second party confirmation size.

CIMPORT_C CCError_t CC_Sm2Kdf (const **CC_Sm2KeContext_t** * *pCtx*, const size_t *SharedSecretSizeInBits*, uint8_t * *pKeyOut*, size_t * *pKeyOutSize*)

Returns:

`CC_OK` on success.

A non-zero value on failure

Parameters:

I/O	Parameter	Description
in	`pCtx`	A Pointer to a key exchange context.
in	`SharedSecretSizeInBits`	The required size of the key in bits.
in	`pKeyOut`	A Pointer to a buffer for the derived key.
in,out	`pKeyOutSize`	A Pointer to the derived key size in bytes.

CIMPORT_C void CC_Sm2KeyExchangeContext_cleanup (**CC_Sm2KeContext_t** * *pCtx*)

Returns:

void.[in] A pointer to a context structure.

CIMPORT_C CCError_t CC_Sm2KeyExchangeContext_init (**CC_Sm2KeContext_t** * *pCtx*, uint8_t * *pWorkingBuffer*, const size_t *wblen*, **CCEcpkiUserPublKey_t** * *pPubKey*, **CCEcpkiUserPrivKey_t** * *pPrivKey*, **CCEcpkiUserPublKey_t** * *pRemoteUserPubKey*, const char * *pId*, size_t *idlen*, const char * *pRemoteId*, size_t *remoteIdLen*, uint8_t *is_initiator*, uint8_t *conf_required*)

Returns:

`CC_OK` on success.

A non-zero value on failure

Parameters:

I/O	Parameter	Description
in,out	`pCtx`	This pointer should be allocated by user. This function inits it.
in	`pWorkingBuffer`	The working buffer
in	`wblen`	The working buffer size should be at least 2 + modSizeInBytes*4 + ordSizeInBytes*2 + max(idlen, ridlen)
in	`pPubKey`	The data of the public key.
in	`pPrivKey`	The data of the private key.
in	`pRemoteUserPubKey`	The data of the remote public key.
in	`pId`	A pointer to the ID.
in	`idlen`	The ID size in bytes.
in	`pRemoteId`	A pointer to an remote ID.
in	`remoteIdLen`	The remote ID size in bytes.
in	`is_initiator`	1 if it is an initiator side.
in	`conf_required`	bit mask - 1st bit if we want conf, 2nd if the other part wants

CIMPORT_C CCError_t CC_Sm2Sign (**CCRndGenerateVectWorkFunc_t** *f_rng*, void * *p_rng*, const **CCEcpkiUserPrivKey_t** * *pSm2PrivKey*, const uint32_t * *pHashInput*, const size_t *HashInputSize*, uint8_t * *pSignatureOut*, size_t * *pSignatureOutSize*)

Algorithm according to the Public key cryptographic algorithm SM2 based on elliptic curves. Part 2: Digital signature algorithm

It takes as an input the message digest as a little endian words that come as an output from the **CC_Sm2ComputeMessageDigest()** function.

Returns:

`CC_OK` on success.

A non-zero value on failure as defined **cc_ecpki_error.h**, **cc_sm3_error.h**, or **cc_rnd_error.h**.

Parameters:

I/O	Parameter	Description
in	`f_rng`	A pointer to DRBG function

I/O	Parameter	Description
in,out	`p_rng`	A pointer to the random context - the input to f_rng.
in	`pSm2PrivKey`	A pointer to a private key structure.
in	`pHashInput`	A pointer to the hash of the input data.
in	`HashInputSize`	The size of message data hash in words.
out	`pSignatureOut`	Pointer to a buffer for output of signature.
in,out	`pSignatureOutSize`	A pointer to the signature size. Used to pass the size of the SignatureOut buffer (in), which must be >= 2 *OrderSizeInBytes. When the API returns, it is replaced with the size of the actual signature (out).

CIMPORT_C CCError_t CC_Sm2Verify (const **CCEcpkiUserPublKey_t** * *pUserPublKey*, uint8_t * *pSignatureIn*, const size_t *SignatureSizeBytes*, const uint32_t * *pHashInput*, const size_t *HashInputSize*)

Algorithm according to the Public key cryptographic algorithm SM2 based on elliptic curves. Part 2: Digital signature algorithm

It takes as an input the message digest as a little endian words that come as an output from the **CC_Sm2ComputeMessageDigest()** function.

Returns:

`CC_OK` on success.

A non-zero value on failure as defined **cc_ecpki_error.h** or **cc_sm3_error.h**.

Parameters:

I/O	Parameter	Description
in	`pUserPublKey`	A pointer to a public key structure.
in	`pSignatureIn`	A pointer to the signature to be verified.
in	`SignatureSizeBytes`	The size of the signature (in bytes).
in	`pHashInput`	A pointer to the hash of the input data that was signed.
in,out	`HashInputSize`	The size of the hash of the input data (in words).

2.6.38.41 cc_sm3.h File Reference

This file contains all the enums and definitions that are used for the CryptoCell SM3 APIs, as well as the APIs themselves.

```
#include "cc_pal_types.h"
```

```
#include "cc_error.h"
```

```
#include "cc_sm3_defs.h"
```

2.6.38.41.1 Functions

- **CIMPORT_C CCError_t CC_Sm3Init** (**CCSm3UserContext_t** *pContextID)

This function initializes the SM3 machine and the SM3 Context.

- **CIMPORT_C CCError_t CC_Sm3Update** (**CCSm3UserContext_t** *pContextID, uint8_t *pDataIn, size_t DataInSize)

  This function processes a block of data to be HASHed.

- **CIMPORT_C CCError_t CC_Sm3Finish** (**CCSm3UserContext_t** *pContextID, **CCSm3ResultBuf_t** Sm3ResultBuff)

  This function finalize the process of SM3 data block.

- **CIMPORT_C CCError_t CC_Sm3Free** (**CCSm3UserContext_t** *pContextID)

  This function frees the context if the operation had failed.

- **CIMPORT_C CCError_t CC_Sm3** (uint8_t *pDataIn, size_t DataInSize, **CCSm3ResultBuf_t** Sm3ResultBuff)

  This function provides an SM3 function to process one buffer of data.

2.6.38.42 cc_sm3_defs.h File Reference

This file contains definitions of the CryptoCell SM3 APIs.

```
#include "cc_pal_types.h"

#include "cc_error.h"

#include "cc_sm3_defs_proj.h"
```

2.6.38.42.1 Data Structures

- struct **CCSm3UserContext_t**

2.6.38.42.2 Macros

- #define **CC_SM3_RESULT_SIZE_IN_BITS** 256
- #define **CC_SM3_RESULT_SIZE_IN_BYTES** (**CC_SM3_RESULT_SIZE_IN_BITS** / **CC_BITS_IN_BYTE**)
- #define **CC_SM3_RESULT_SIZE_IN_WORDS** (CC_SM3_RESULT_SIZE_IN_BYTES / **CC_32BIT_WORD_SIZE**)
- #define **CC_SM3_BLOCK_SIZE_IN_BYTES** 64
- #define **CC_SM3_BLOCK_SIZE_IN_WORDS** 16
- #define **CC_SM3_UPDATE_DATA_MAX_SIZE_IN_BYTES** (1 << 61)

2.6.38.42.3 Typedefs

- typedef uint8_t **CCSm3ResultBuf_t**[CC_SM3_RESULT_SIZE_IN_BYTES]
- typedef struct **CCSm3UserContext_t CCSm3UserContext_t**

2.6.38.43 cc_sm3_defs_proj.h File Reference

This file contains SM3 definitions.

2.6.38.43.1 Macros

- #define CC_SM3_USER_CTX_SIZE_IN_WORDS 165

2.6.38.44 cc_sm3_error.h File Reference

This file contains the definitions of the CryptoCell SM3 errors.

```
#include "cc_error.h"
```

2.6.38.44.1 Macros

- #define CC_SM3_INVALID_USER_CONTEXT_POINTER_ERROR (CC_SM3_MODULE_ERROR_BASE + 0x0UL)
- #define CC_SM3_USER_CONTEXT_CORRUPTED_ERROR (CC_SM3_MODULE_ERROR_BASE + 0x1UL)
- #define CC_SM3_DATA_IN_POINTER_INVALID_ERROR (CC_SM3_MODULE_ERROR_BASE + 0x2UL)
- #define CC_SM3_DATA_SIZE_ILLEGAL (CC_SM3_MODULE_ERROR_BASE + 0x3UL)
- #define CC_SM3_INVALID_RESULT_BUFFER_POINTER_ERROR (CC_SM3_MODULE_ERROR_BASE + 0x4UL)
- #define CC_SM3_LAST_BLOCK_ALREADY_PROCESSED_ERROR (CC_SM3_MODULE_ERROR_BASE + 0x5UL)
- #define CC_SM3_ILLEGAL_PARAMS_ERROR (CC_SM3_MODULE_ERROR_BASE + 0x6UL)
- #define CC_SM3_CTX_SIZES_ERROR (CC_SM3_MODULE_ERROR_BASE + 0x7UL)
- #define CC_SM3_IS_NOT_SUPPORTED (CC_SM3_MODULE_ERROR_BASE + 0x8UL)

2.6.38.45 cc_sm4.h File Reference

This file contains all the enums and definitions that are used for the CryptoCell SM4 APIs, as well as the APIs themselves.

```
#include "cc_pal_types.h"

#include "cc_sm4_defs.h"
```

2.6.38.45.1 Functions

- **CIMPORT_C CCError_t CC_Sm4Init** (**CCSm4UserContext_t** *pContext, **CCSm4EncryptMode_t** encryptDecryptFlag, **CCSm4OperationMode_t** operationMode)

  This function is used to initialize a SM4 operation context. To operate the SM4 machine, this must be the first API called.

- **CIMPORT_C CCError_t CC_Sm4SetKey** (**CCSm4UserContext_t** *pContext, **CCSm4Key_t** pKey)

  This function sets the key information for the SM4 operation, in the context that was initialized by **CC_Sm4Init()**.

- **CIMPORT_C CCError_t CC_Sm4SetIv** (**CCSm4UserContext_t** *pContext, **CCSm4Iv_t** pIV)

  This function sets the IV or counter data for the following SM4 operations on the same context. The context must be first initialized by **CC_Sm4Init()**. It must be called at least once prior to the first **CC_Sm4Block()** operation on the same context - for those ciphers that require it. If needed, it can also be called to override the IV in the middle of a sequence of **CC_Sm4Block()** operations.

- **CIMPORT_C CCError_t CC_Sm4GetIv** (**CCSm4UserContext_t** *pContext, **CCSm4Iv_t** pIV)

  This function retrieves the current IV or counter data from the SM4 context.

- **CIMPORT_C CCError_t CC_Sm4Block** (**CCSm4UserContext_t** *pContext, uint8_t *pDataIn, size_t dataSize, uint8_t *pDataOut)

  This function performs a SM4 operation on an input data buffer, according to the configuration defined in the context parameter. It can be called as many times as needed, until all the input data is processed. The functions **CC_Sm4Init()**, **CC_Sm4SetKey()**, and for some ciphers **CC_Sm4SetIv()**, must be called before the first call to this API with the same context.

- **CIMPORT_C CCError_t CC_Sm4Finish** (**CCSm4UserContext_t** *pContext, uint8_t *pDataIn, size_t dataSize, uint8_t *pDataOut)

  This function is used to finish SM4 operation. It processes the last data block if needed, and finalizes the SM4 operation (cipher-specific).

- **CIMPORT_C CCError_t CC_Sm4Free** (**CCSm4UserContext_t** *pContext)

  This function releases and clears resources after SM4 operations.

- **CIMPORT_C CCError_t CC_Sm4** (**CCSm4Iv_t** pIV, **CCSm4Key_t** pKey, **CCSm4EncryptMode_t** encryptDecryptFlag, **CCSm4OperationMode_t** operationMode, uint8_t *pDataIn, size_t dataSize, uint8_t *pDataOut)

  This function performs a SM4 operation with a given key in a single call for all SM4 supported modes, and can be used when all data is available at the beginning of the operation.

2.6.38.46 cc_sm4_defs.h File Reference

This file contains the type definitions that are used by the CryptoCell SM4 APIs.

```
#include "cc_sm4_defs_proj.h"
```

2.6.38.46.1 Data Structures

- struct CCSm4UserContext_t

2.6.38.46.2 Macros

- #define CC_SM4_CRYPTO_BLOCK_SIZE_IN_WORDS 4
- #define CC_SM4_BLOCK_SIZE_IN_BYTES (CC_SM4_CRYPTO_BLOCK_SIZE_IN_WORDS *sizeof(uint32_t))
- #define CC_SM4_KEY_SIZE_IN_WORDS CC_SM4_CRYPTO_BLOCK_SIZE_IN_WORDS
- #define CC_SM4_KEY_SIZE_IN_BYTES (CC_SM4_KEY_SIZE_IN_WORDS *sizeof(uint32_t))
- #define CC_SM4_IV_SIZE_IN_WORDS CC_SM4_CRYPTO_BLOCK_SIZE_IN_WORDS
- #define CC_SM4_IV_SIZE_IN_BYTES (CC_SM4_IV_SIZE_IN_WORDS *sizeof(uint32_t))

2.6.38.46.3 Typedefs

- typedef uint8_t CCSm4Iv_t[CC_SM4_IV_SIZE_IN_BYTES]
- typedef uint8_t CCSm4Key_t[CC_SM4_KEY_SIZE_IN_BYTES]
- typedef struct CCSm4UserContext_t CCSm4UserContext_t

2.6.38.46.4 Enumerations

- enum CCSm4EncryptMode_t { CC_SM4_ENCRYPT = 0, CC_SM4_DECRYPT = 1, CC_SM4_NUM_OF_ENCRYPT_MODES, CC_SM4_ENCRYPT_MODE_LAST = 0x7FFFFFFF }
- enum CCSm4OperationMode_t { CC_SM4_MODE_ECB = 0, CC_SM4_MODE_CBC = 1, CC_SM4_MODE_CTR = 2, CC_SM4_MODE_OFB = 3, CC_SM4_NUM_OF_OPERATION_MODES, CC_SM4_OPERATION_MODE_LAST = 0x7FFFFFFF }

2.6.38.47 cc_sm4_defs_proj.h File Reference

This file contains definitions that are used in the CryptoCell SM4 APIs.

2.6.38.47.1 Macros

- #define CC_SM4_USER_CTX_SIZE_IN_WORDS 131

2.6.38.48 cc_sm4_error.h File Reference

This file contains the definitions of the CryptoCell SM4 errors.

```
#include "cc_error.h"
```

2.6.38.48.1 Macros

- #define CC_SM4_INVALID_USER_CONTEXT_POINTER_ERROR (CC_SM4_MODULE_ERROR_BASE + 0x00UL)
- #define CC_SM4_INVALID_IV_POINTER_ERROR (CC_SM4_MODULE_ERROR_BASE + 0x01UL)
- #define CC_SM4_ILLEGAL_OPERATION_MODE_ERROR (CC_SM4_MODULE_ERROR_BASE + 0x02UL)
- #define CC_SM4_ILLEGAL_KEY_SIZE_ERROR (CC_SM4_MODULE_ERROR_BASE + 0x03UL)
- #define CC_SM4_INVALID_KEY_POINTER_ERROR (CC_SM4_MODULE_ERROR_BASE + 0x04UL)
- #define CC_SM4_INVALID_ENCRYPT_MODE_ERROR (CC_SM4_MODULE_ERROR_BASE + 0x05UL)
- #define CC_SM4_USER_CONTEXT_CORRUPTED_ERROR (CC_SM4_MODULE_ERROR_BASE + 0x06UL)
- #define CC_SM4_DATA_IN_POINTER_INVALID_ERROR (CC_SM4_MODULE_ERROR_BASE + 0x07UL)
- #define CC_SM4_DATA_OUT_POINTER_INVALID_ERROR (CC_SM4_MODULE_ERROR_BASE + 0x08UL)
- #define CC_SM4_DATA_IN_SIZE_ILLEGAL (CC_SM4_MODULE_ERROR_BASE + 0x09UL)
- #define CC_SM4_ILLEGAL_PARAMS_ERROR (CC_SM4_MODULE_ERROR_BASE + 0x0AUL)
- #define CC_SM4_ILLEGAL_INPLACE_ERROR (CC_SM4_MODULE_ERROR_BASE + 0x0BUL)
- #define CC_SM4_IS_NOT_SUPPORTED (CC_SM4_MODULE_ERROR_BASE + 0xFFUL)

2.6.38.49 cc_trng_error.h File Reference

This file contains the definitions of the CryptoCell TRNG errors.

```
#include "cc_error.h"
```

2.6.38.49.1 Macros

- #define CC_TRNG_INVALID_PARAMS_ERROR (CC_TRNG_MODULE_ERROR_BASE + 0x0UL)

2.6.38.50 cc_trng_fe.h File Reference

This file contains API and definitions for generating TRNG buffer in full entropy mode.

```
#include "cc_pal_types.h"
```

2.6.38.50.1 Macros

- #define **CC_TRNG_MIN_ENTROPY_SIZE** 0
- #define **CC_TRNG_MAX_ENTROPY_SIZE** 8192

2.6.38.50.2 Functions

- **CCError_t CC_TrngEntropyGet** (size_t entropySizeBits, uint8_t *pOutEntropy, size_t outEntropySizeBytes)

  The function returns an entropy buffer in the requested size.

2.6.38.51 cc_util_error.h File Reference

This file contains the error definitions of the CryptoCell utility APIs.

2.6.38.51.1 Macros

- #define **CC_UTIL_OK** 0x00UL
- #define **CC_UTIL_MODULE_ERROR_BASE** 0x80000000
- #define **CC_UTIL_INVALID_KEY_TYPE** (**CC_UTIL_MODULE_ERROR_BASE** + 0x00UL)
- #define **CC_UTIL_DATA_IN_POINTER_INVALID_ERROR** (**CC_UTIL_MODULE_ERROR_BASE** + 0x01UL)
- #define **CC_UTIL_DATA_IN_SIZE_INVALID_ERROR** (**CC_UTIL_MODULE_ERROR_BASE** + 0x02UL)
- #define **CC_UTIL_DATA_OUT_POINTER_INVALID_ERROR** (**CC_UTIL_MODULE_ERROR_BASE** + 0x03UL)
- #define **CC_UTIL_DATA_OUT_SIZE_INVALID_ERROR** (**CC_UTIL_MODULE_ERROR_BASE** + 0x04UL)
- #define **CC_UTIL_FATAL_ERROR** (**CC_UTIL_MODULE_ERROR_BASE** + 0x05UL)
- #define **CC_UTIL_ILLEGAL_PARAMS_ERROR** (**CC_UTIL_MODULE_ERROR_BASE** + 0x06UL)
- #define **CC_UTIL_BAD_ADDR_ERROR** (**CC_UTIL_MODULE_ERROR_BASE** + 0x07UL)
- #define **CC_UTIL_EK_DOMAIN_INVALID_ERROR** (**CC_UTIL_MODULE_ERROR_BASE** + 0x08UL)
- #define **CC_UTIL_KDR_INVALID_ERROR** (**CC_UTIL_MODULE_ERROR_BASE** + 0x09UL)

- #define **CC_UTIL_KCP_INVALID_ERROR** (**CC_UTIL_MODULE_ERROR_BASE** + 0x0AUL)
- #define **CC_UTIL_KPICV_INVALID_ERROR** (**CC_UTIL_MODULE_ERROR_BASE** + 0x0BUL)
- #define **CC_UTIL_KCST_NOT_DISABLED_ERROR** (**CC_UTIL_MODULE_ERROR_BASE** + 0x0CUL)
- #define **CC_UTIL_LCS_INVALID_ERROR** (**CC_UTIL_MODULE_ERROR_BASE** + 0x0DUL)
- #define **CC_UTIL_SESSION_KEY_ERROR** (**CC_UTIL_MODULE_ERROR_BASE** + 0x0EUL)
- #define **CC_UTIL_INVALID_USER_KEY_SIZE** (**CC_UTIL_MODULE_ERROR_BASE** + 0x0FUL)
- #define **CC_UTIL_ILLEGAL_LCS_FOR_OPERATION_ERR** (**CC_UTIL_MODULE_ERROR_BASE** + 0x10UL)
- #define **CC_UTIL_INVALID_PRF_TYPE** (**CC_UTIL_MODULE_ERROR_BASE** + 0x11UL)
- #define **CC_UTIL_INVALID_HASH_MODE** (**CC_UTIL_MODULE_ERROR_BASE** + 0x12UL)
- #define **CC_UTIL_UNSUPPORTED_HASH_MODE** (**CC_UTIL_MODULE_ERROR_BASE** + 0x13UL)
- #define **CC_UTIL_KEY_UNUSABLE_ERROR** (**CC_UTIL_MODULE_ERROR_BASE** + 0x14UL)
- #define **CC_UTIL_PM_ERROR** (**CC_UTIL_MODULE_ERROR_BASE** + 0x15UL)
- #define **CC_UTIL_SD_IS_SET_ERROR** (**CC_UTIL_MODULE_ERROR_BASE** + 0x16UL)

2.6.38.51.2 Typedefs

- typedef uint32_t **CCUtilError_t**

3 Integration Test API layer

3.1 Modules

Here is a list of all modules:

- HAL board integration tests
- PAL integration tests
- PAL memory integration tests
- PAL thread integration tests
- PAL timer functions
- Test definitions and APIs

3.2 File list

The following table lists the files that are part of the delivery, and their descriptions:

Table 3-1 List of files

Filename	Description
`board_configs.h`	This file contains board initialization functions.
`test_pal_map_addrs.h`	This file contains PAL map address integration tests.
`test_pal_mem.h`	This file contains PAL memory integration tests.
`test_pal_thread.h`	This file contains the PAL thread integration tests.
`test_pal_time.h`	This file contains PAL time functions.
`test_proj_plat.h`	This file contains definitions and APIs that set the testing environment.

3.3 Module Documentation

3.3.1 HAL board integration tests

Contains HAL board integration functions.

3.3.1.1 Functions

- uint32_t **Test_HalBoardInit** (void)

  This function initializes the board.

- void **Test_HalBoardFree** (void)

  This function unmaps the addresses related to the board.

3.3.1.2 Function documentation

3.3.1.2.1 void Test_HalBoardFree (void)

Returns:

void

3.3.1.2.2 uint32_t Test_HalBoardInit (void)

Returns:

0 on success.

1 on failure.

3.3.2 PAL integration tests

Contains PAL map address integration tests.

3.3.2.1 Macros

- #define **VALID_MAPPED_ADDR**(addr) ((addr != 0) && (addr != 0xFFFFFFFF))
- #define **BM_READ** 0x01
- #define **BM_WRITE** 0x02
- #define **BM_EXEC** 0x04
- #define **BM_NONE** 0x08
- #define **BM_SHARED** 0x10
- #define **BM_PRIVATE** 0x20
- #define **BM_FIXED** 0x40

3.3.2.2 Functions

- void ***Test_PalIOMap** (void *physAddr, size_t size)

  This function maps IO physical address to OS accessible address.

- void ***Test_PalMapAddr** (void *physAddr, void *startingAddr, const char *filename, size_t size, uint8_t protAndFlagsBitMask)

  This function maps a physical address to a virtual address.
- void **Test_PalUnmapAddr** (void *virtAddr, size_t size)

  This function unmaps a virtual address.

3.3.2.3 Macro definition documentation

3.3.2.3.1 #define BM_EXEC 0x04

Pages can be executed.

3.3.2.3.2 #define BM_FIXED 0x40

Do not interpret address as a hint: place the mapping at exactly that address.

3.3.2.3.3 #define BM_NONE 0x08

Pages cannot be accessed.

3.3.2.3.4 #define BM_PRIVATE 0x20

Create a private copy-on-write mapping.

3.3.2.3.5 #define BM_READ 0x01

Pages can be read.

3.3.2.3.6 #define BM_SHARED 0x10

Share this mapping.

3.3.2.3.7 #define BM_WRITE 0x02

Pages can be written.

3.3.2.4 Function documentation

3.3.2.4.1 void*Test_PalIOMap (void * *physAddr*, size_t *size*)

Returns:

A valid virtual address.

Null on case of failure.

Parameters:

Parameter	Description
physAddr	Physical address.
size	Size in bytes.

3.3.2.4.2 void*Test_PalMapAddr (void * *physAddr*, void * *startingAddr*, const char * *filename*, size_t *size*, uint8_t *protAndFlagsBitMask*)

Returns:

A valid virtual address

Null on failure.

Parameters:

Parameter	Description
physAddr	A physical address.
startingAddr	Preferred static address for mapping.
filename	File name.
size	Size in bytes.
protAndFlagsBitMask	Protection and update visibility bit mask.

3.3.2.4.3 void Test_PalUnmapAddr (void * *virtAddr*, size_t *size*)

Returns:

Parameters:

Parameter	Description
virtAddr	Virtual address.Size in bytes.

3.3.3 PAL memory integration tests

Contains PAL memory integration tests.

3.3.3.1 Functions

- void *Test_PalMalloc (size_t size)

  This function allocates "size" bytes. When TZM is supported, it is used only for NON SECURE memory allocations.

- void Test_PalFree (void *pvAddress)

This function frees allocated memory pointed by pvAddress. When TZM is supported, it is used only for NON SECURE memory blocks.

- void ***Test_PalRealloc*** (void *pvAddress, size_t newSize)

  This function changes the size of the memory block pointed by pvAddress. If the function fails to allocate the requested block of memory:

- void ***Test_PalDMAContigBufferAlloc*** (size_t size)

  This function allocates a DMA-contiguous buffer and returns its address. When TZM is supported, it is used only for NON SECURE buffer allocations.

- void **Test_PalDMAContigBufferFree** (void *pvAddress)

  This function frees resources previously allocated by Test_PalDMAContigBufferAlloc. When TZM is supported, it is used only for NON SECURE buffers.

- void ***Test_PalDMAContigBufferRealloc*** (void *pvAddress, size_t newSize)

  This function changes the size of the memory block pointed by pvAddress. If the function fails to allocate the requested block of memory:

- unsigned long **Test_PalGetDMABaseAddr** (void)

  This function returns DMA base address, i.e. the start address of the DMA region. When TZM is supported, it returns the NON SECURE DMA base address.

- unsigned long **Test_PalGetUnmanagedBaseAddr** (void)

  This function returns the unmanaged base address. When TZM is supported, it returns the NON SECURE unmanaged base address.

- uint32_t **Test_PalMemInit** (unsigned long newDMABaseAddr, unsigned long newUnmanagedBaseAddr, size_t DMAsize)

  This function initializes DMA memory management. When TZM is supported, it initializes the NON SECURE DMA memory management.

- uint32_t **Test_PalMemFin** (void)

  This function sets this driver to its initial state. When TZM is supported, it sets the NON SECURE management to its initial state.

3.3.3.2 Function documentation

3.3.3.2.1 void*Test_PalDMAContigBufferAlloc (size_t *size*)

Returns:

Address of the allocated buffer.

NULL on failure.

Parameters:

Parameter	Description
size	Buffer size in bytes.

3.3.3.2.2 void Test_PalDMAContigBufferFree (void * *pvAddress*)

Returns:

void

Parameters:

Parameter	Description
pvAddress	Address of the allocated buffer.

3.3.3.2.3 void*Test_PalDMAContigBufferRealloc (void * *pvAddress*, size_t *newSize*)

- A null pointer is returned.
- The memory block pointed by argument pvAddress is NOT deallocated.

When TZM is supported, it is used only for NON SECURE buffers.

Returns:

A pointer to the new allocated memory.

Parameters:

Parameter	Description
pvAddress	Pointer to the allocated memory.
newSize	New size in bytes.

3.3.3.2.4 void Test_PalFree (void * *pvAddress*)

Returns:

void

Parameters:

Parameter	Description
pvAddress	Pointer to the allocated memory.

3.3.3.2.5 unsigned long Test_PalGetDMABaseAddr (void)

Returns:

DMA base address

3.3.3.2.6 unsigned long Test_PalGetUnmanagedBaseAddr (void)

Returns:

Unmanaged base address.

3.3.3.2.7 void*Test_PalMalloc (size_t *size*)

Returns:

Pointer to the allocated memory.

NULL on failure.

Parameters:

Parameter	Description
`size`	Size in bytes.

3.3.3.2.8 uint32_t Test_PalMemFin (void)

Returns:

0 on success

1 on failure.

3.3.3.2.9 uint32_t Test_PalMemInit (unsigned long *newDMABaseAddr*, unsigned long *newUnmanagedBaseAddr*, size_t *DMAsize*)

Returns:

0 on success

1 on failure.

Parameters:

Parameter	Description
`newDMABaseAddr`	New DMA start address.
`newUnmanagedBaseAddr`	New unmanaged start address.
`DMAsize`	DMA region size.

3.3.3.2.10 void*Test_PalRealloc (void * *pvAddress*, size_t *newSize*)

- A null pointer is returned.
- The memory block pointed by argument pvAddress is NOT deallocated.

When TZM is supported, it is used only for NON SECURE memory blocks.

Returns:

A pointer to the new allocated memory on success.

NULL on failure.

Parameters:

Parameter	Description
`pvAddress`	Pointer to the allocated memory.
`newSize`	New size.

3.3.4 PAL thread integration tests

Contains the PAL thread integration tests.

3.3.4.1 typedefs

- typedef void ***ThreadHandle**

3.3.4.2 Functions

- size_t **Test_PalGetMinimalStackSize** (void)

  This function returns the minimal stack size in bytes.

- uint32_t **Test_PalGetHighestPriority** (void)

  This function returns the highest thread priority.

- uint32_t **Test_PalGetLowestPriority** (void)

  This function returns the lowest thread priority.

- uint32_t **Test_PalGetDefaultPriority** (void)

  This function returns the default thread priority.

- **ThreadHandle Test_PalThreadCreate** (size_t stackSize, void *(*threadFunc)(void *), int priority, void *args, const char *threadName, uint8_t nameLen, uint8_t dmaAble)

  This function creates a thread. The user should call **Test_PalThreadJoin()** in order to wait until the thread ends and then to **Test_PalThreadDestroy()** in order to free resources. In case of a thread without an end, the user should not call **Test_PalThreadJoin()** which will not return. Instead, the user should call **Test_PalThreadDestroy()** which will cancel the thread and free its resources.

- uint32_t **Test_PalThreadJoin** (**ThreadHandle** threadHandle, void **threadRet)

  This function waits for a thread to terminate (BLOCKING). If that thread has already terminated it returns immediately.

- uint32_t **Test_PalThreadDestroy** (**ThreadHandle** threadHandle)

This function destroys a thread (if it's still running) and frees its resources. In order to free thread resources only after thread's end this function should be called after **Test_PalThreadJoin()**. In order to cancel the thread immediately and free its resources, this function should be called alone (without **Test_PalThreadJoin()**), which must eventually be called in any case. Note that this function does not deallocate the memory that the thread itself allocates. This needs to be done by the thread itself.

3.3.4.3 typedef documentation

3.3.4.3.1 typedef void*ThreadHandle

Thread handle

3.3.4.4 Function documentation

3.3.4.4.1 uint32_t Test_PalGetDefaultPriority (void)

Returns:

Default thread priority.

3.3.4.4.2 uint32_t Test_PalGetHighestPriority (void)

Returns:

Highest thread priority.

3.3.4.4.3 uint32_t Test_PalGetLowestPriority (void)

Returns:

Lowest thread priority.

3.3.4.4.4 size_t Test_PalGetMinimalStackSize (void)

Returns:

Minimal stack size in bytes.

3.3.4.4.5 ThreadHandle Test_PalThreadCreate (size_t *stackSize*, void *(*)(void *) *threadFunc*, int *priority*, void * *args*, const char * *threadName*, uint8_t *nameLen*, uint8_t *dmaAble*)

Returns:

Thread handle address on success

NULL on failure.

Parameters:

Parameter	Description
`stackSize`	Thread stack size in bytes. The allocated stack size will be greater from stackSize and the minimal stack size.
`threadFunc`	Thread function. The function shall return a pointer to the returned value or NULL. In case TZM is supported, this function must have the same security attribute as TestAL's (either secure or non-secure).
`priority`	Thread priority. Highest and lowest priorities can be received by calling **Test_PalGetLowestPriority()** and **Test_PalGetHighestPriority()** accordingly.
`args`	Function input arguments.
`threadName`	Thread name. Not in use for Linux.
`nameLen`	Thread name length. Not in use for Linux.
`dmaAble`	Determines whether the stack should be DMA-able (true).

3.3.4.4.6 uint32_t Test_PalThreadDestroy (ThreadHandle *threadHandle*)

Returns:

0 on success

1 on failure.

Parameters:

Parameter	Description
`threadHandle`	Thread structure.

3.3.4.4.7 uint32_t Test_PalThreadJoin (ThreadHandle *threadHandle*, void ** *threadRet*)

- threadRet is not changed, yet `threadRet` is changed and can be NULL. Therefore, do not try to access `threadRet` without checking that `threadRet` is not NULL.

Returns:

0 on success

1 on failure.

Parameters:

Parameter	Description
`threadHandle`	Thread structure.
`threadRet`	A pointer to the returned value of the target thread.

3.3.5 PAL timer functions

Contains PAL timer functions.

3.3.5.1 Functions

- void **Test_PalDelay** (const uint32_t usec)

  This function suspends execution of the calling thread for microsecond intervals.

- uint32_t **Test_PalGetTimestamp** (void)

  This function returns a timestamp in milliseconds.

3.3.5.2 Function documentation

3.3.5.2.1 void Test_PalDelay (const uint32_t *usec*)

Returns:

Void

Parameters:

Parameter	Description
usec	Time to suspend in microseconds.

3.3.5.2.2 uint32_t Test_PalGetTimestamp (void)

Returns:

Timestamp in milliseconds.

3.3.6 Test definitions and APIs

Contains definitions and APIs that set the testing environment.

3.3.6.1 Macros

- #define **TEST_READ_ENV_REG**(offset) *(volatile uint32_t *)(processMap.processTeeHwEnvBaseAddr + (offset))
- #define **TEST_WRITE_ENV_REG**(offset, val)

3.3.6.2 typedefs

- typedef enum **TestProjCache_t TestProjCache_t**

3.3.6.3 Enumerations

- enum **TestProjCache_t** { **TEST_PROJ_HW_CACHE**, **TEST_PROJ_SW_CACHE** }

3.3.6.4 Functions

- uint32_t **Test_ProjMap** (void)

  This function maps the CryptoCell base register and environment base register.

- void **Test_ProjUnmap** (void)

  This function unmaps the CryptoCell base register and environment base register.

- void **Test_ProjPerformPowerOnReset** (void)

  This function performs power-on-reset to CryptoCell, AO & environment modules using environment register.

- void **Test_ProjPerformColdReset** (void)

  This function performs cold-reset to CryptoCell and AO modules using environment register.

- void **Test_ProjPerformWarmReset** (void)

  This function performs warm-reset to CryptoCell module using environment register.

- void **Test_ProjSetSpEnable** (void)

  This function sets the Sp_enable bit to CryptoCell module.

- void **Test_ProjSetCacheParams** (**TestProjCache_t** cacheType)

  This function sets the cache parameters. The set operation is done via environment registers.

- void **Test_ProjSetSecureMode** (void)

  This function sets the device security mode. The set operation is done via environment registers.

3.3.6.5 Macro definition documentation

3.3.6.5.1 #define TEST_READ_ENV_REG(offset) *(volatile uint32_t *)(processMap.processTeeHwEnvBaseAddr + (offset))

Defines Environment register read.

You must implement the read environment register that is compatible with your system.

3.3.6.5.2 #define TEST_WRITE_ENV_REG(offset, val)

```
{ \
        volatile uint32_t ii1; \
        (*(volatile uint32_t *)(processMap.processTeeHwEnvBaseAddr + (offset)))
= (uint32_t)(val); \
        for(ii1=0; ii1<500; ii1++); \
}
```

Defines Environment register write.

You must implement the write environment register that is compatible with your system.

3.3.6.6 typedef documentation

3.3.6.6.1 typedef enum TestProjCache_t TestProjCache_t

Defines the cache parameters group set for the environment register.

3.3.6.7 Enumeration type documentation

3.3.6.7.1 enum TestProjCache_t

Defines the cache parameters group set for the environment register.

Enumerator:

Enum	Description
`TEST_PROJ_HW_CACHE`	AxUSER - HW - 1.
`TEST_PROJ_SW_CACHE`	AxUSER - HW - 0.

3.3.6.8 Function documentation

3.3.6.8.1 uint32_t Test_ProjMap (void)

You must replace the environment mapping with implementation that is compatible with your system.

Returns:

`TEST_OK` on success.

A non-zero value from test_proj_common.h on failure.

3.3.6.8.2 void Test_ProjPerformColdReset (void)

You must define cold-reset implementation that is compatible with your system.

Returns:

`Void`

3.3.6.8.3 void Test_ProjPerformPowerOnReset (void)

You must define power-on-reset implementation that is compatible with your system.

Returns:

`Void`

3.3.6.8.4 void Test_ProjPerformWarmReset (void)

You must define warm-reset implementation that is compatible with your system.

Returns:

`Void`

3.3.6.8.5 void Test_ProjSetCacheParams (TestProjCache_t *cacheType*)

You must replace TEST_READ_OTP_BY_ENV() macro with implementation that is compatible with your system.

Returns:

`Void`

3.3.6.8.6 void Test_ProjSetSecureMode (void)

You must replace TEST_READ_OTP_BY_ENV() macro with implementation that is compatible with your system.

Returns:

`Void`

3.3.6.8.7 void Test_ProjSetSpEnable (void)

Returns:

`Void`

3.3.6.8.8 void Test_ProjUnmap (void)

You must replace the environment un-mapping with implementation that is compatible with your system.

Returns:

```
Void.
```

3.4 File Documentation

3.4.1 board_configs.h file reference

This file contains board initialization functions.

3.4.1.1 Functions

- uint32_t **Test_HalBoardInit** (void)

  This function initializes the board.

- void **Test_HalBoardFree** (void)

  This function unmaps the addresses related to the board.

3.4.2 test_pal_map_addrs.h file reference

This file contains PAL map address integration tests.

3.4.2.1 Macros

- #define **VALID_MAPPED_ADDR**(addr) ((addr != 0) && (addr != 0xFFFFFFFF))
- #define **BM_READ** 0x01
- #define **BM_WRITE** 0x02
- #define **BM_EXEC** 0x04
- #define **BM_NONE** 0x08
- #define **BM_SHARED** 0x10
- #define **BM_PRIVATE** 0x20

- #define **BM_FIXED** 0x40

3.4.2.2 Functions

- void ***Test_PalIOMap*** (void *physAddr, size_t size)

  This function maps IO physical address to OS accessible address.

- void ***Test_PalMapAddr*** (void *physAddr, void *startingAddr, const char *filename, size_t size, uint8_t protAndFlagsBitMask)

  This function maps a physical address to a virtual address.

- void **Test_PalUnmapAddr** (void *virtAddr, size_t size)

  This function unmaps a virtual address.

3.4.3 test_pal_mem.h file reference

This file contains PAL memory integration tests.

```
#include <stdint.h>

#include <stdio.h>
```

3.4.3.1 Functions

- void ***Test_PalMalloc*** (size_t size)

  This function allocates "size" bytes. When TZM is supported, it is used only for NON SECURE memory allocations.

- void **Test_PalFree** (void *pvAddress)

  This function frees allocated memory pointed by pvAddress. When TZM is supported, it is used only for NON SECURE memory blocks.

- void ***Test_PalRealloc*** (void *pvAddress, size_t newSize)

  This function changes the size of the memory block pointed by pvAddress. If the function fails to allocate the requested block of memory:

- void ***Test_PalDMAContigBufferAlloc*** (size_t size)

  This function allocates a DMA-contiguous buffer and returns its address. When TZM is supported, it is used only for NON SECURE buffer allocations.

- void **Test_PalDMAContigBufferFree** (void *pvAddress)

  This function frees resources previously allocated by Test_PalDMAContigBufferAlloc. When TZM is supported, it is used only for NON SECURE buffers.

- void ***Test_PalDMAContigBufferRealloc*** (void *pvAddress, size_t newSize)

This function changes the size of the memory block pointed by pvAddress. If the function fails to allocate the requested block of memory:

- unsigned long **Test_PalGetDMABaseAddr** (void)

  This function returns DMA base address, i.e. the start address of the DMA region. When TZM is supported, it returns the NON SECURE DMA base address.

- unsigned long **Test_PalGetUnmanagedBaseAddr** (void)

  This function returns the unmanaged base address. When TZM is supported, it returns the NON SECURE unmanaged base address.

- uint32_t **Test_PalMemInit** (unsigned long newDMABaseAddr, unsigned long newUnmanagedBaseAddr, size_t DMAsize)

  This function initializes DMA memory management. When TZM is supported, it initializes the NON SECURE DMA memory management.

- uint32_t **Test_PalMemFin** (void)

  This function sets this driver to its initial state. When TZM is supported, it sets the NON SECURE management to its initial state.

3.4.4 test_pal_thread.h file reference

This file contains the PAL thread integration tests.

```
#include <stdint.h>
```

3.4.4.1 typedefs

- typedef void ***ThreadHandle**

3.4.4.2 Functions

- size_t **Test_PalGetMinimalStackSize** (void)

  This function returns the minimal stack size in bytes.

- uint32_t **Test_PalGetHighestPriority** (void)

  This function returns the highest thread priority.

- uint32_t **Test_PalGetLowestPriority** (void)

  This function returns the lowest thread priority.

- uint32_t **Test_PalGetDefaultPriority** (void)

  This function returns the default thread priority.

- **ThreadHandle Test_PalThreadCreate** (size_t stackSize, void *(*threadFunc)(void *), int priority, void *args, const char *threadName, uint8_t nameLen, uint8_t dmaAble)

  This function creates a thread. The user should call **Test_PalThreadJoin()** in order to wait until the thread ends and then to **Test_PalThreadDestroy()** in order to free resources. In case of a thread without an end, the user should not call **Test_PalThreadJoin()** which will not return. Instead, the user should call **Test_PalThreadDestroy()** which will cancel the thread and free its resources.

- uint32_t **Test_PalThreadJoin** (**ThreadHandle** threadHandle, void **threadRet)

  This function waits for a thread to terminate (BLOCKING). If that thread has already terminated it returns immediately.

- uint32_t **Test_PalThreadDestroy** (**ThreadHandle** threadHandle)

  This function destroys a thread (if it's still running) and frees its resources. In order to free thread resources only after thread's end this function should be called after **Test_PalThreadJoin()**. In order to cancel the thread immediately and free its resources, this function should be called alone (without **Test_PalThreadJoin()**), which must eventually be called in any case. Note that this function does not deallocate the memory that the thread itself allocates. This needs to be done by the thread itself.

3.4.5 test_pal_time.h file reference

This file contains PAL time functions.

```
#include <stdint.h>
```

3.4.5.1 Functions

- void **Test_PalDelay** (const uint32_t usec)

  This function suspends execution of the calling thread for microsecond intervals.

- uint32_t **Test_PalGetTimestamp** (void)

  This function returns a timestamp in milliseconds.

3.4.6 test_proj_plat.h file reference

This file contains definitions and APIs that set the testing environment.

```
#include <stdint.h>
```

3.4.6.1 Macros

- #define **TEST_READ_ENV_REG**(offset) *(volatile uint32_t *)(processMap.processTeeHwEnvBaseAddr + (offset))
- #define **TEST_WRITE_ENV_REG**(offset, val)

3.4.6.2 typedefs

- typedef enum **TestProjCache_t TestProjCache_t**

3.4.6.3 Enumerations

- enum **TestProjCache_t** { **TEST_PROJ_HW_CACHE**, **TEST_PROJ_SW_CACHE** }

3.4.6.4 Functions

- uint32_t **Test_ProjMap** (void)

  This function maps the CryptoCell base register and environment base register.

- void **Test_ProjUnmap** (void)

  This function unmaps the CryptoCell base register and environment base register.

- void **Test_ProjPerformPowerOnReset** (void)

  This function performs power-on-reset to CryptoCell, AO & environment modules using environment register.

- void **Test_ProjPerformColdReset** (void)

  This function performs cold-reset to CryptoCell and AO modules using environment register.

- void **Test_ProjPerformWarmReset** (void)

  This function performs warm-reset to CryptoCell module using environment register.

- void **Test_ProjSetSpEnable** (void)

  This function sets the sp enable bit to CryptoCell module.

- void **Test_ProjSetCacheParams** (**TestProjCache_t** cacheType)

  This function sets the cache parameters. The set operation is done via environment registers.

- void **Test_ProjSetSecureMode** (void)

  This function sets the device security mode. The set operation is done via environment registers.

- void **Test_ProjSetFlavor** (void)

This function sets the fpga to slim/full mode according to CC_SUPPORT_FULL_PROJECT flag. The set operation is done via environment registers. This function is needed for testing with FPGA.

- void **Test_ProjSetFullFlavor** (void)

  This function resets the FPGA to the original flavor it was in - full The set operation is done via environment registers. This function is needed for testing with FPGA.

3.4.6.5 Function documentation

3.4.6.5.1 void Test_ProjSetFlavor (void)

Returns:

```
None
```

3.4.6.5.2 void Test_ProjSetFullFlavor (void)

Returns:

```
None
```

Appendix A Revisions

Table A-1 Issue 0000-01

Change	Location	Affects
This is the first release of this product	-	-